(2)

NPS-MA-92-009

NAVAL POSTGRADUATE SCHOOL

Monterey, California

INVESTIGATION OF BATTLE TRACE DISPLAYS
FOR TRAINING APPLICATIONS

Donald Barr
James Hoffman

August 1992

Prepared for: Naval Postgraduate School and
TRADOC Analysis Command, Monterey,
CA 93943

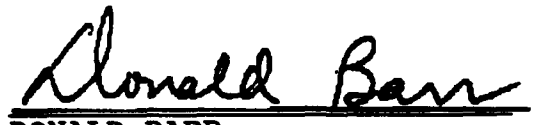
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

Rear Admiral R. W. West, Jr.
Superintendent

Harrison Shull
Procost

This report was prepared in conjunction with research conducted for the Naval Postgraduate School and TRADOC Analysis Command. Funding was provided by the Naval Postgraduate School and TRADOC Analysis Command.

Prepared by:



DONALD BARR
Adjunct Professor of Mathematics

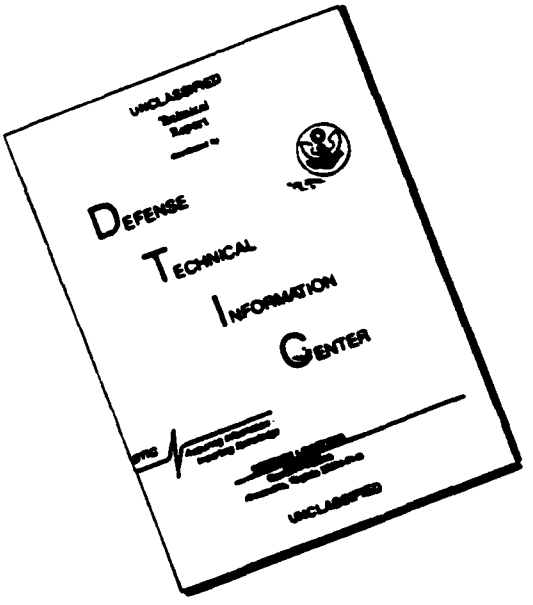
Reviewed by:

RICHARD FRANKE
Chairman
Department of Mathematics

Released by:

PAUL J. MARTO
Dean of Research

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release: distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-MA-92-009	5 MONITORING ORGANIZATION REPORT NUMBER(S) NPS_MA-92-009
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) MA	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943
8a NAME OF FUNDING/SPONSORING ORGANIZATION TRADOC Analysis Command 8b OFFICE SYMBOL (If applicable) TRAC	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code) Naval Postgraduate School Monterey, CA 93943	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Investigation of Battle Trace Displays for Training Applications

12 PERSONAL AUTHOR(S)
Donald Barr, James Hoffman

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical Report	FROM 7/91 TO 6/92	August 7, 1992	54

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Battle Trace, Janus, Combat Simulations, Training, Measures of Effectiveness

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The battle trace is a plot representing success of a combatant as a function of time into a battle. The battle trace can be driven by combat simulations such as Janus(A), providing a real-time indicator of combat success. Methods of exploiting this technology in a training context such as the National Training Center are examined.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION
22a NAME OF RESPONSIBLE INDIVIDUAL Donald Barr	22b TELEPHONE (Include Area Code) (914) 938-3078 22c OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete.*

S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

INVESTIGATION OF BATTLE TRACE DISPLAYS FOR TRAINING APPLICATIONS

Jim Hoffman
TRADOC Analysis Command-Monterey

and

Donald R. Barr
Naval Postgraduate School

April 1992

Abstract

The battle trace is a plot representing success of a combatant as a function of time into a battle. The battle trace can be driven by combat simulations such as Janus(A), providing a real-time indicator of combat success. Methods of exploiting this technology in a training context such as the National Training Center are examined.

Key Words

Battle Trace
Janus
Combat Simulation
Training
Measures of Effectiveness

1. Introduction

The "battle trace" is a high level indicator of success in a battle, computed as a function of time into the battle. Barr, Weir and Hoffman [1991] derive the battle trace based on conditions in a battle assumed to follow the Lanchester Square Law. Suppose combatants X and Y engage in a battle and suppose the battle is partitioned into portions corresponding to time increments Δ_1, Δ_2, Δ_3,..., which together cover the duration of the battle. In the i^{th} time interval, the battle trace is defined to be $B=[\Delta y/y]/[\Delta x/x]$, where Δy and Δx are measures of casualties sustained by Y and X, respectively, within the given time interval, and y and x are measures of the strengths of Y and X, respectively, during that period.

Barr, Weir and Hoffman [1991] also consider two symmetric variants of B, also refered to as "battle trace measures:"

the "symmetric battle trace,"

$$B_S = \begin{cases} B, & \text{if } B \geq 1; \\ 2-1/B, & \text{if } B<1, \end{cases}$$

and the "log trace,"

$$B_L = \ln(\Delta y+\alpha) + \ln(x+\alpha) - \ln(y+\alpha) - \ln(\Delta x+\alpha),$$

where α is a suitable positive constant. (Battle traces with arguments increased by such α are called "incremented" in what follows.) They discuss how these measures might be used in applications related to field testing and combat simulation models.

Battle Trace Displays

In the present paper we discuss possible uses of battle traces in combat training environments. Specifically, we envision that the Janus combat simulation model might be used to enhance training conducted at the National Training Center (NTC) at Fort Irwin, CA. Once a unit has completed a set of battles at the NTC, the relevant parameters of the battles (such as scenarios, routes followed by forces on each side, uses of scouting platoons, engineering activities, etc.) can be set (with varying degrees of fidelity) into the Janus combat simulation, and the battles can be replayed within Janus. The objective of such replays is to provide the unit commander information about how the battle might have progressed had various changes been made by the commander in his prosecution of the battle. For example, a commander might wish to examine how his forces would have done had he used his scouting assets more effectively, or if he had exploited terrain masking more effectively in maneuvering his forces toward an attack.

However, there is considerable uncontrolled variation in battle outcomes, both for battles at the NTC and those simulated within Janus. Thus, even if one replicates a battle by holding the controllable factors at the same levels while running a second battle, generally there will be substantial differences between the two battles nonetheless. Any attempt to exactly reproduce the results of a given battle is doomed to failure, so by extension there is not a one-to-one relationship between the

set of values of parameters in a battle simulation and consequent ways in which the simulated battles unfold over time. It may be difficult, therefore, for a battle commander to interpret effects on battle results that might be due to changes in battle parameters or conditions.[1]

With the substantial variations in battle outcomes, how can a commander get a clear picture of the likely effects of changing selected battle parameters? Estimates based on traditional measures of effectiveness (MOE's) generally provide definitive information about battle outcomes only when it is feasible to carry out numerous replications with each combination of factors under study. We doubt this is feasible in the context of using Janus to augment NTC training as mentioned above. Thus it is desirable to employ alternative measures that give general, "high-level" views of the progress of each battle while atttempting to avoid the volatility associated with specific end-of-battle MOE's. We believe battle traces may be useful in this regard.

Several questions about computation of battle trace measures were investigated:

---

[1]This is precisely why force-on-force field tests generally are designed to employ many replications within each cell of the test matrix. Such "variance reduction by replication" is not feasible in most training applications at the present time because of resource constraints. Eventually it may be possible to run a significant number of Janus replications simultaneously using parallel computing techniques, so meaningful sample sizes can be attained in "real time."

Battle Trace Displays

* How should the time increments Δt_i be chosen?
* Which version of the battle trace is best for training applications?
* Should the battle trace plot be smoothed? If so, what is the best method of smoothing?

Numerous crude simulation experiments were conducted to provide rough initial answers to these questions. These experiments provided information relevant to making a choice of which battle trace measure should be used in the initial demonstrations of the concept. The resulting choice is described below. Whether this particular measure does indeed help a commander understand the general effects of making certain changes in how he fights a given battle is, at this point, an open question. This issue can be examined by future experimentation. An indicator of the utility of the measure will be the degree to which it is readily accepted and used.

In the sections that follow, we describe properties of the selected form of the battle trace and document how it was implemented in the Janus(A) simulation at TRAC-Monterey. We expect demonstrations with this form of the battle trace may lead to performance of experiments to evaluate its utility in the training context.

Battle Trace Displays

2. Initial Simulation Results

As mentioned above, early in our investiagtion we generated various battle traces with data obtained from crude simulations. The simulated battles followed the Lanchester Square Law within each time interval, Δt_i; i=1,2,...,n, in a partition of the battle period. This was done to generate data required in computation of the battle trace in its various forms. The general approach was to examine the appearance and stability of various forms of battle trace displays computed with the generated data. Several related mathematical developments are presented in Appendices 1-3. In Section 6 we report results of preliminary experiments with the selected battle trace when it is run from within Janus.

We drew the following general conclusions from these initial experiments:

(1) The plots of the various battle trace functions have considerable variation (i.e., are not smooth). This is aggravated when the time sub-intervals Δt are too small. We found that, for the simulated battles we considered, plots with 20 to 50 data points for the entire battle (lasting about 40 minutes) looked best.

(2) Potential division by zero in B and B_S, and zero arguments within log functions in B_L, cause undesirable and unnecessary problems in the computations. It appears the "incremented" versions of the battle trace

functions successfully avoid these problems without significant loss of fidelity.

(3) Plots using dynamic definitions of Δt appear to have advantages over those with periods of fixed length. It appears definitions involving Δx or Δy being sufficiently large (or attaining sufficiently large values of other quantities related to expected casualties, such as the number of shots fired or received by one or both forces) work well. The values of thresholds defining "sufficiently large" should be chosen so as to provide an incremented battle trace with a fairly smooth plot. Choice of such a threshold generally involves the expected length and outcome of the battle and the numbers of combatants involved. It appears that not more than about 50 data points should be generated over the duration of the battle; fewer data points would be appropriate if there are relatively few combatants involved.

As a result of our initial experiments we concluded that, on balance, an incremented version of the log trace, B_L is a good choice. Points on the log trace should be computed with Δt defined dynamically. For runs conducted within Janus(A) at TRAC-Monterey we implemented a rule involving the number of shots fired by or received by each task force under consideration. The purpose of computing battle trace in Janus(A) was to conduct

demonstrations of the concept and to provide further experiments and evaluations of the utility of battle traces. Details of this implementation are given in the Section 6.

3. Discussion of the Recommended Battle Trace Form

Even though we are recommending the log trace B_L, we discuss computation of B here, since its form admits an interesing and relatively simple interpretation. Since B_L is simply related to B, this discussion is relevant to B_L as well. We will use the term "battle trace" in a generic sense to refer to any form of the measures under discussion.

The numbers of casualties occuring on each side during an interval of time can be considered to be outcomes on random variables. The means of these random variables, that is, the expected numbers of casualties in the time interval, can be estimated by summing the p_k's associated with shots upon each force during the time interval. Such estimators, called "sums of p_k's" and denoted "Σp_k," are discussed in Appendix 2. There, it is shown that use of appropriate Σp_k's to represent the expected casualties Δx and Δy, will reduce the variance of computed values of points on battle traces, and hence will make more precise (and possibly appear to "smooth") their plots. We used this technique in forming the log trace discussed below.

In a similar way, the numbers of relevant combatants involved in the battle at a given time (i.e., values of y and x

in the formula for B_L) can be considered to be outcomes on random quantities. It is desirable to use variance reducing estimators for the means of the distributions of x and y within each corresponding battle time interval. By analogy with the discussion in Appendix 2, it appears that an estimator similar to the sum of p_k's estimator would be appropriate for this purpose. Consider estimating a quantity to represent x in the battle trace computation. Within the Lanchester Square Law, x denotes the size of the X force at the time in question; note, however, the effect of these X forces is to decrease y by approximately $b \cdot x$. The effect of x in the Lanchester model is thus related to the number of units in the X force that may engage Y units. One possibility for the battle trace computation is thus to count, in the i^{th} time interval, the number of X units that have line of sight (or, alternatively, have detected or have acquired) at least one unit of the Y force. This would correspond roughly to the body count estimator (as discussed in Appendix 2) of the mean of the x distribution.

As an alternative, we propose summing an appropriate measure of the potential of each X unit to inflict casualties on the Y force. Due to the way in which the Janus simulation processes information and updates target lists, it is feasible to determine, for each unit of the X force, the maximum probability of kill, maxp, within the i^{th} time interval, against any target which is detectable by the unit (i.e., is intervisible, is within

the unit's sector of regard, and is within its maximum range limits). Let $maxp_j$ denote the value of maxp for the j^{th} X-unit in the i^{th} time interval and suppose there are N X-units alive at the beginning of this time interval. We propose using the value $\Sigma_{j=1}^{N} maxp_j$ to estimate the mean of $k \cdot x$, and to use this in computation of the battle trace. Note this value is a measure of the potential of the X force to inflict casualties during the i^{th} time interval. A similar estimator, summing maxp over Y-units, measures the potential of the Y force to inflict casualties on X. This is used to estimate the mean of $k \cdot y$. In the battle trace computation, the unknown proportionality constant k, assumed to be the same in numerator and denominator of B, cancels.

If the sum of p_k's estimators are used for Δx and Δy and the sum of maxp's are used for x and y as described above, the battle trace can be interpreted in the following way. Let px_k denote the probability of kill by an X-unit (against a Y-unit) in the k^{th} engagement, with a similar definition for py_k; let $maxpx_j$ denote the maximum probability of kill by the j^{th} X-unit against any of its Y-targets during the i^{th} time interval, with a similar definition for $maxpy_j$. Then for the i^{th} time interval, the battle trace

$$B=[\Delta y/y]/[\Delta x/x]$$

is computed as

Battle Trace Displays

$$B \approx \frac{\sum_{\text{shots on Y}} px}{\sum_{\text{Y-units}} maxpy} \Big/ \frac{\sum_{\text{shots on X}} py}{\sum_{\text{X-units}} maxpx}$$

$$= \frac{\sum_{\text{shots on Y}} px}{\sum_{\text{shots on X}} py} \times \frac{\sum_{\text{X-units}} maxpx}{\sum_{\text{Y-units}} maxpy} . \qquad (6)$$

The first of the ratios in equation (6) is the expected casualty exchange ratio, and is a measure of the relative firing effectiveness of the X and Y forces during the i^{th} time interval. The numerator (denominator, respectively) is a sum of single shot probabilities of kill for all shots taken by the X force at the Y force (all shots by the Y force on the X force) in the given time interval. These terms account for the influence of weapon firing during an interval of battle which includes both direct and indirect (artillery) round impacts. Because there are typically many more shots fired than kills, such values contain more information about the related battle processes than does a simple count of casualties.

The second of the ratios in equation (6) is a force capabilities ratio; it measures the relative attrition potential of the X and Y forces. The quantity in numerator of this ratio is the sum of the maxumum values of single shot probability of kill for each X weapon system which can detect at least one system of the opposing (Y) force some time during the given time interval. Conditions on detectability include existence of line

Battle Trace Displays

of sight between the combatants, as well as conformance with bounds on the sector of regard (the X-unit's search sector) and range.

Similar comments hold for the denominator of the second ratio in (6). Thus these sums contain information on the potential of a force to influence the outcome of battle irrespective of weapon firing events and independent of rate of fire. These values may be seen to reflect the mass of force on each side of the conflict generated within each time interval of the battle. A weapon system which has run out of ammunition or has no ammunition suitable for engaging detectable targets can make little direct contribution to the battle. This fact is reflected in the second ratio in equation (6) since then the corresponding maxp is zero, hence such a system is not counted in the expressions corresponding to x or y in the formula for B. Selecting the maximum single shot probability of kill value as the potential contribution for a given weapon system is consistent with an assumption that each weapon's crew will work to maximize its contribution to the battle.

In terms of the two ratios in equation (6) above, the X force is winning the battle at a given point in the battle if it is engaging targets more effectively than is Y, and it is maneuvering, positioning, using cover and concealment and in general is preparing for battle more effectively than is its adversary. These conditions will generate relatively high values

Battle Trace Displays

of the casualty exchange ratio (first part of equation 6) and the attrition potential ratio (second part of equation 6), so the battle trace B will be significantly larger than 1.0. If the incremented log trace is being used (as we recommend), B_L will then be significantly larger than zero.

On the other hand, X may be losing at a given point in the battle even though it enjoys a superior casualty exchange at that point (first ratio in (6) greater than 1.0), if Y is out-maneuvering and out-planning X so as to have sufficiently high advantage in attrition potential (second ratio in (6) much smaller than 1.0). It is then possible that the product B of the two ratios is less than 1.0, so the log trace would be less than zero.

4. Implementation Within Janus(A) at TRAC-Monterey

In Janus simulations of battle, forces on each side may be partitioned into task forces by the modeler. The compositions of these task forces can be varied by a force commander during the play of a battle. We next consider computation of a battle trace for each of possibly several task forces on each side, where the task force compositions at each point of the battle are consistent with any changes made by the force commanders during Janus execution. To simplify our discussion, first we consider computation of the battle trace for the j^{th} Blue Task Force, B_j, which we consider to be on side X. Let us begin by describing

how each of the sums in equation (6) are computed for this task force.

The sum of p_k's corresponding to the first term in the battle trace, Δy, is

$$\sum_{\substack{\text{all red who can see } B_j \\ \text{but are shot at by} \\ \text{any Blue unit}}} P_{k(B-R)}$$

where "$P_{k(B-R)}$" denotes a probability of kill by a Blue firer upon a Red target. Similarly, the sum of p_k's corresponding to the third term of B, Δx, is

$$\sum_{\substack{\text{all units} \\ \in B_j}} P_{k(R-B_j)}$$

The sum of $maxp_k$'s corresponding to the second term of B, y, is

$$\sum_{\substack{\text{all Red units} \\ \text{for which at least one} \\ \text{unit} \in B_j \text{ is detectable}}} maxP_{k(R-B)}$$

and the sum of $maxp_k$'s corresponding to the fourth term of B, x, is

$$\sum_{\substack{\text{units} \in B_j \text{ who} \\ \text{could detect any} \\ \text{red unit}}} maxP_{k(B-R)}$$

A separate battle trace should be computed for each task force of interest on the Blue side and each task force of interest on the Red side. Additionally, combinations of task Battle Trace Displays

forces could be considered (all Blue units, for example) or pseudo task forces might be defined (all Blue tanks, for example). In the following section we describe how the quantities in equation (6) were captured within the Janus(A) simulation for each task force on each side.

5. Details of Implementation

This section contains a description of how the Janus(A) code was modified at TRAC-Monterey so experiments and demonstrations with the battle trace could be carried out. Specific listings of Fortran code comprising major subroutines of Janus(A) that were so modified are shown in Appendix 5. Together with the comments in this section, the listings in Appendix 5 should serve as documentation of our implementation of battle trace in Janus.

With the revised Janus code, several arrays are maintained throughout a simulation of a battle, to provide the data required to support computation of battle traces at the end of appropriate time intervals. These are defined as follows, where "numsides" is the number of sides (1=Blue and 2=Red), "numtasks" is the number of task forces on a given side and "numunits" is the number of units on a given side:[2]

$SSHOTPKS_{(numsides)x(numtasks)}$ - the ij^{th} element is the sum of p_k's of all shots by side i against all elements of side 3-i

[2]The dimensions of each array are shown as subscrpts to the array name.

who could detect at least one unit in task force j of side i. The elements of this array are updated just before each call to subroutine SHOT within subroutine RELOAD of Janus. The update involves adding the p_k value to be used in SHOT to certain components of the row of SSHOTPKS corresponding to the side of the firing unit. The components that are incremented are those corresponding to side i task forces detectable by the target fired upon.

$SIPCTPKS_{(numsides)x(numtasks)}$ - the ij^{th} element is the sum of p_k's of shots fired at the task force j of side i (by any firer on side 3-i). The elements of this array are updated just prior to each shot by adding the p_k for the shot called in subroutine SHOT, as described above, to the appropriate element.

$BTRACEVAL_{(numsides)x(numunits)x(numtasks)}$ - the ijk^{th} element is the maximum of all p_k values for the j^{th} unit on the i^{th} side against any detectable units of the k^{th} task force of the $(3-i)^{th}$ (opposing) side. This array is updated each time a target list is determined within Janus (in subroutine DETECT of Janus).

$BTINFL_{(numsides)x(numunits)x(numtasks)}$ - the ijk^{th} element is a flag indicating whether the j^{th} unit on side i could detect the k^{th} task force on the $(3-i)^{th}$ side. This array is updated each time a target list is determined, as described above.

$KNUMSHOTS_{(numsides)x(numtasks)}$ - the ij^{th} element is the number

of shots fired by or on the j^{th} task force of side i.

$BTMAX_{(numsides)x(numunits)}$ - the ij^{th} element is the maximum over columns (opposing task forces) of BTRACEVAL (i.e., the max-reduction of BTRACEVAL across columns). This reduction is calculated just prior to computing a point on the battle trace for the j^{th} task force of side i.

$PKUNITMAX_{(numsides)x(numtasks)}$ - the ij^{th} element is the sum of elements in the j^{th} column of BTRACEVAL, giving the sum over units on side 3-i of max p_ks against task force j of side i. Elements of this array are updated when BTMAX is updated.

$PKTASKMAX_{(numsides)x(numtasks)}$ - the ij^{th} element is the sum of BTMAX(i,k) over k (units) in taskforce j.

The battle traces are computed using values in the arrays described above. At each cycle of Janus (every 20 seconds of battle time), computation of a point on the battle trace for task force JTASK on side JSIDE is undertaken if the condition KNUMSHOTS(JSIDE,JTASK) $\geq$ t, where the threshold t is set as an input parameter in the Janus set-up. Use of this rule provides dynamic allocation of the lengths of the time intervals over which the battle traces are computed; the times will occur as discrete multiples of the 20 second cycle time inherent in the Janus update schedule. Points will be computed more frequently for a task force when it is heavily engaged with the enemy (either firing or receiving fires).

When threshold conditions are met for one or more task

Battle Trace Displays

forces, the corresponding points on the battle trace plots are computed and re-initialization of appropriate elements of the arrays described above takes place.

When a threshold condition is met, BTMAX and PKUNITMAX are updated and the components needed for computation of the corresponding battle trace point are extracted from the arrays described above. Following the battle trace computation appropriate elements of the arrays are set to zero in preparation for accumulating values in the next time interval for the succeeding point on that battle trace. In terms of the quantities in the arrays described above, the battle trace value shown in equation (6), for task force j of side i, becomes:

$$B = \frac{SSHOTPKS(i,j)}{SIPCTPKS(i,j)} \times \frac{PKTASKMAX(i,j)}{PKUNITMAX(i,j)}$$

Immediately after computing the point on the battle trace, the following array elements are set to zero:

SSHOTPKS(i,j);

SIPCTPKS(i,j);

BTMAX(i,j);

BTRACEVAL(i,j,k) for all task forces k on side 3-i.

The following pseudo-code illustrates how this is implemented; specific listings of the fortran code are given in Appendix 5.

Battle Trace Displays

Pseudo-code for computing log trace

```
**comment: check each battle trace threshold condition**

for i=1 to numsides
     for j=1 to numtasks
          if KNUMSHOTS(i,j) < t then go to GOON

          **comment: if condition met, compute B_L **

          R=0
          for k=1 to numunits        **comment:sum column of
                                     BTRACEVAL & initialize**
               R=R+BTRACEVAL(3-i,k,j)
               BTRACEVAL(3-i,k,j)=0
          next k
          B=0
          for L=1 to numunits    **comment:sum maxpk for
                                 units in taskforce &
                                 initialize**
               if L in taskforce(i,j) then B=B+MAXPK(i,L)
               if L in taskforce(i,j) then MAXPK(i,L)=0
          next L
     B_L(i,j)=ln(SSHOTPKS(i,j) + 1) - ln(SIPCTPKS(i,j) + 1)
                                     -ln(R + 1) + ln(B + 1)
     SSHOTKPKS(i,j) = 0
     SIPCTPKS(i,j) = 0
GOON next j
next i
```

6. Results of Preliminary Experiments with Janus(A)

To gain a tentative impression of battle traces obtained with the modified Janus(A) simulation, we conducted limited experiments at TRAC-Monterey. These experiments all relate to battles simulated with a "Fulda Gap" scenario developed at TRAC-Monterey for use with a Janus tutorial. We obtained intuitive confirmation of the indications gained with the earlier crude

Battle Trace Displays

simulation study described in Section 2: the incremented log trace with dynamic definition of Δt appears to be a good candidate for more extensive testing in a training context.

We observed that there may be considerable variation in specific points on battle traces for iterations of the same battle. However, the general features of battle traces for such iterations appear to be quite similar, suggesting the battle traces are capable of capturing major events and trends of a battle. Plots of battle traces for iterations of the Fulda Gap scenario are shown in Appendix 4. We believe the variations seen in battle traces for iterations of a battle simply show how different the specific details of battles can be, even though the controllable parameters are held constant (see Figure 3 of Appendix 4).

It was observed that the log trace tended to have a shape similar to a plot of casualty exchange ratios (CER) calculated within each time interval (Figure 5). This is to be expected since the CER is one of two ratios involved in computation of the battle trace (see equation (6)). But there is theoretical support for this relationship, based on the Lanchester model. This can be seen as follows: assuming

$$dB/dt = -\beta R \quad \text{and} \quad dR/dt = -\alpha B$$

it follows that

Battle Trace Displays

$$\frac{B}{R} = \frac{\frac{dR}{dt} / -\alpha}{\frac{dB}{dt} / -\beta} \approx \frac{\beta}{\alpha} \times \frac{\Delta R}{\Delta B}$$

Thus

$$Battle\ trace = \frac{\Delta R}{\Delta B} \times \frac{B}{R} \approx \frac{\beta}{\alpha} \times (\frac{\Delta R}{\Delta B})^2$$

so

$$B_L = \ln(\frac{\beta}{\alpha}) + 2\ln(\frac{\Delta R}{\Delta B}) = c_0 + c_1 \ln(CER) \quad .$$

If the Lanchester model fits reasonable well, a plot of log trace versus ln(CER) should appear as a line with slope $c_1 = 2$. If this appears tenable, one can estimate the ratio (α/β) by e^{c_0}. We show a plot of B_L versus ln(CER) in Figure 6, together with a line with slope 2. It appears the line fits the data from the Fulda Gap scenario rather well.

Residuals representing differences between plotted points of the form $(B_L, \ln(CER))$ from the line with slope 2 show variations in (β/α) at the end of each time interval of battle. That is, variations of the plotted points from the line indicate variations of the battle away from a Lanchester square-law battle. Indeed, one can get rough estimates of $\ln(\beta/\alpha)$, within time increments making up the battle period, using the values (residual + c_0), where c_0 is obtained from the fit of the line with slope 2. A plot of these values versus time might serve as an indicator of how well Red is doing relative to Blue; we have

Battle Trace Displays

plotted such curves in Figure 7 for seven iterations of the Fulda Gap scenario.

In a masters thesis written at the Air Force Institute of Technology, H.K. Choi suggested a variant form of the battle trace which he thought might have greater stability than has B_L. Mr. Choi suggested that since Blue wins in a Lanchester square law battle whenever $C = \beta R^2 - \alpha B^2 < 0$, it might be useful to plot values of

$$C = \frac{\frac{-dB}{dt}}{R} \times R^2 - \frac{\frac{-dR}{dt}}{B} \times B^2 \approx \frac{\Delta B \cdot R - \Delta R \cdot B}{\Delta t}$$

within each time interval versus time into the battle. We think this suggestion has potential merit, and we have plotted this form of battle trace, which we denote "B_C," for seven iterations of the Fulda Gap scenario, in Figure 9.

7. References

Barr, D.R., M. Weir, and J. Hoffman (1991), "Evaluation of Combat," Naval Postgraduate School Technical Report NPS-MA-92-001, 23 pp.

Choi, H.K. (1992), "Analysis of Measures of Combat Performance," Masters Thesis, Air Force Institute of Technology.

James Hoffman and D.R. Barr (1992), "An indicator of combat success," submitted for publication.

Appendix 1. A Time Transformation in the Lanchester Square Law

In the initial crude simulations used to generate data for the battle trace experiments described in Section 2, we varied only one attrition parameter, a(t), in the Lanchester Square Law. The following argument shows this can be done without loss of generality.

Consider a transformation T=f(t) of the time parameter in the Lanchester Square Law

$$dx = -a(t)\cdot y(t)\ dt; \qquad (7)$$

$$dy = -b(t)\cdot x(t)\ dt.$$

Assume f is 1-1 and differentiable so that $dT = f'(t)dt$ and $t = f^{-1}(T)$. Substituting for t and dt in equations (7) gives

$$d[x(f^{-1}(T))] = -a(f^{-1}(T))\cdot y(f^{-1}(T))\cdot dT/f'(t) \qquad (8)$$

$$= dx(f^{-1}(T))/f'(t)$$

(assuming $f'(t) \neq 0$) and

$$d[y(f^{-1}(T))] = -b(f^{-1}(T))\cdot x(f^{-1}(T))\cdot dT/f'(t) \qquad (9)$$

$$= dy(f^{-1}(T))/f'(t).$$

Letting

$$X(T) = x(f^{-1}(T)),$$

$$Y(T) = y(f^{-1}(T)),$$

$$A(T) = a(f^{-1}(T))/f'(f^{-1}(T)),$$

$$B(T) = b(f^{-1}(T))/f'(f^{-1}(T)),$$

it follows that

$$dX(T) = -A(T)\cdot Y(T)\ dT; \qquad (10)$$

Battle Trace Displays

$$dY(T) = -B(T) \cdot X(T)\ dT,$$

which is just the Lanchester Square Law with a new time parameter T and "transformed" attrition coefficients.

Now suppose

$$f(t) = \alpha \cdot \int_0^t b(\tau)\, d\tau \qquad (11)$$

so $f'(t) \equiv \alpha b(t)$. Then $f'(f^{-1}(T) \equiv \alpha b(f^{-1}(T)$, so $B(T) \equiv \alpha$. With the transformation defined in equation (11) the attrition coefficient B in equations (10) is made constant. This implies one can assume without loss of generality that b is constant in equations (7).

Battle Trace Displays

Appendix 2. Variance Reduction

We consider here a method of computing Δy and Δx within a time interval Δt_i which provides a reduction in the variance of these statistics. In the Lanchester Square Law, these quantities represent the numbers of casualties occurring on sides Y and X, respectively, during the time interval in question. But within the simulated battles (whether at the NTC or with the Janus combat simulation), the numbers of casualties are outcomes depending on numerous random events and conditions occurring as the battle evolves in time. We thus consider determination of Δy and Δx as a statistical estimation problem. We suppose there are true (but unknown) mean values of the distributions of numbers of casualties on each side within the i^{th} time interval, and we consider the task of estimating these means. The estimates will then be used as input values of Δy and Δx in the computation of the battle trace.

An obvious way to estimate the expected number of battle units of a certain type that would be killed in a future battle with identical conditions is to count the number of units of this type that were killed in the simulated battle that is under consideration. This estimator, known as the body count estimator, is not commonly used in practice because a superior estimator, the "sum of p_k's" estimator, is available. Suppose M engagements occur in the i^{th} time interval, and the k^{th} of these has kill probability p_k. A sum of p_k's estimate is formed by

summing the p_k's used in the simulated battle over all engagements against targets of the type in question.

The body count and sum of p_k's estimators can be compared as follows. Suppose results of engagements against targets of the type in question are represented by Bernoulli random variables $X_1, X_2, \ldots, X_M$. Here, X_k is zero if the kth engagement against a target of the given type does not result in a kill (which will happen with probability $1 - p_k$, say), and X_k is 1 if the kth engagement does result in a kill. The random variable M is the number of engagements against the type of target in question in the simulated battle. The number of casualties of the given type is $\Sigma_{k=1}^{M} X_k$ so the body count estimator is $\hat{C} = \Sigma_{k=1}^{M} X_k$. The sum of p_k's estimator is $C^* = \Sigma_{k=1}^{M} p_k$. Since the expected value of X_k is p_k, both $\hat{C}$ and C^* have expected value $E[\Sigma_{k=1}^{M} p_k]$, where "E" denotes expectation with respect to the distribution of M. This is the correct mean (that is, both $\hat{C}$ and C^* are unbiased estimators). However, the variance of $\hat{C}$ is

$$E[\Sigma_{k=1}^{M} p_k(1-p_k)] + V[\Sigma_{n=1}^{M} p_k],$$

where "V" denotes variance with respect to the distribution of M. The variance of C^* is equal to only the second term in this expression and is thus smaller than the variance of $\hat{C}$. In statistical terminology, the sum of p_k's estimator is <u>better</u> than the body count estimator. As mentioned above, the kth term in the sum, p_k, is the expected number of kills on the kth engagement, since the expected value of X_k is p_k. Thus the sum

Battle Trace Displays

of p_k's estimator can be interpreted as summing fractional expected kills resulting from each engagement.

Appendix 3. Approximate Variance of B_L

It may be desirable to plot an indicator of the standard error of the battle trace along with the trace values themselves. In what follows we consider how the variance of the log trace might be approximated.

As indicated in Section 1, the log trace is defined by

$$B_L=[\ln(\Delta y+\alpha) + \ln(x+\alpha)] - [\ln(\Delta x+\alpha) + \ln(y+\alpha)], \qquad (12)$$

where α is a suitable positive constant. The specific value of α has relatively little effect on our considerations here, so we ignore it in the remainder of this Appendix. Assume the quantities associated with force sizes in the second Lanchester Square Law equations (7) are random; these equations suggest $-\Delta y$ and x have positive covariance, and similarly for $-\Delta x$ and y.

Now let us consider the variance of $[\ln(\Delta y) + \ln(x)]$. Take the log of both sides of the second equation in (7) to obtain

$$\ln(-\Delta y) = \ln(b) + \ln(x) + \ln(\Delta t),$$

or, assigning simplifying symbols,

$$Y = U + X + k,$$

where k is constant (or at worst is not correlated with X). Assume temporarily that the random variables have been centered at their means, so

$$\mathrm{Cov}(Y,X) = E(Y \cdot X)$$

$$= E([U + X + k]X)$$

$$= E(UX + X^2 + kX)$$

$$= V(X),$$

assuming U and X are uncorrelated.

It follows that

$$V(Y + X) = V(Y) + V(X) + 2Cov(Y,X)$$

$$= V(Y) + 3V(X)$$

$$= V(U) + V(X) + 3V(X)$$

$$= V(U) + 4V(X).$$

With a similar argument for the second term in expression (7), the variance of B_L is, under our assumptions,

$$V(B_L) = 4[V(\ln(x)) + V(\ln(y))] + [V(\ln(b)) + V(\ln(a))]. \quad (13)$$

Since we anticipate that ln(b) and ln(a) will have small variances relative to those of ln(x) and ln(y), it may reasonable to approximate $V(B_L)$ by the first term in expression (13).

The variances of ln(x) and ln(y) can be estimated through a sample of their respective values in the preceding few time intervals (as determined by the sum of maxp estimators). As an alternative, since we plan to use the estimated variances to establish a measure of the quality of the estimated battle trace, it may suffice to use a quantity that only "tracks" the underlying standard deviation. The variances of ln(x) and ln(y) can be roughly bounded as follows. Following the argument for the variance of the sum of p_k's estimator C* above, it can be seen that the sum over X-units of maxpx (see Appendix 2) has

variance approximately equal to

$$E_N[\Sigma_{X\text{-units}} \text{maxpx} \mid N] \approx N \cdot \text{maxpx}, \qquad (14)$$

where "E_N" denotes the expectation with respect to the random number N of X-units involved in the i^{th} time interval. Since x will tend to be greater than 1, ln(x) will have variance less than that of x. Thus a bounding value may be obtained through expression (14), as follows. The averages on the right-hand side of (14) can either be estimated within the current time interval by using the observed values n and maxpx, or a pooled estimate can be maintained, combining with data from previous time intervals. It might be noted that the first of these estimators suggests that, very roughly, the standard deviation of B_L may parallel (fluctuate with) $2\sqrt{(x + y)}$, or for our purposes, simply $\sqrt{(x + y)}$. Some experimentation will be necessary to see whether this value gives a useful indication of the variation seen in B_L.

Appendix 4. Plots of Log Trace for Janus Runs

Example log traces were made with Janus runs using the implementation described in Section 5 and documented in Appendix 5. The runs all used a "Fulda Gap" scenario developed at TRAC-Monterey as part of a Janus Tutorial. In this scenario a Blue force defends against a numerically superior Red attacker. The tutorial concerns placement of a Blue task force (which we call "task force Blue 1") consisting primarily of tanks and armored personnel carriers overlooking a river crossing that is likely to

be used by the attacker. We consider also a part of the Red force that eventually engages with Blue 1 to be a Red task force ("Red 1"). We ran seven iterations of this scenario; several of the battle trace plots that follow show plots for all seven runs superimposed on one figure. The intent of these displays is to illustrate how battle traces can vary from one iteration of a battle to another.

Most of the figures relate to the Blue 1 task force or to the Red force. In some cases the Red 1 task force is considered. These figures suggest Blue 1 enjoyed success early in each battle, but for times beyond about two minutes the Red force is generally winning. In several of the figures it can be seen that Blue 1 suffers particular difficulties around time t=4 minutes. This corresponds to a point in the scenario at which surviving units of Blue task force 1 are set in motion for the first time in the battle. The relevant battle traces suggest this movement is initiated too far into the battle, and it proves disastrous for Blue 1.

Battle Trace Displays

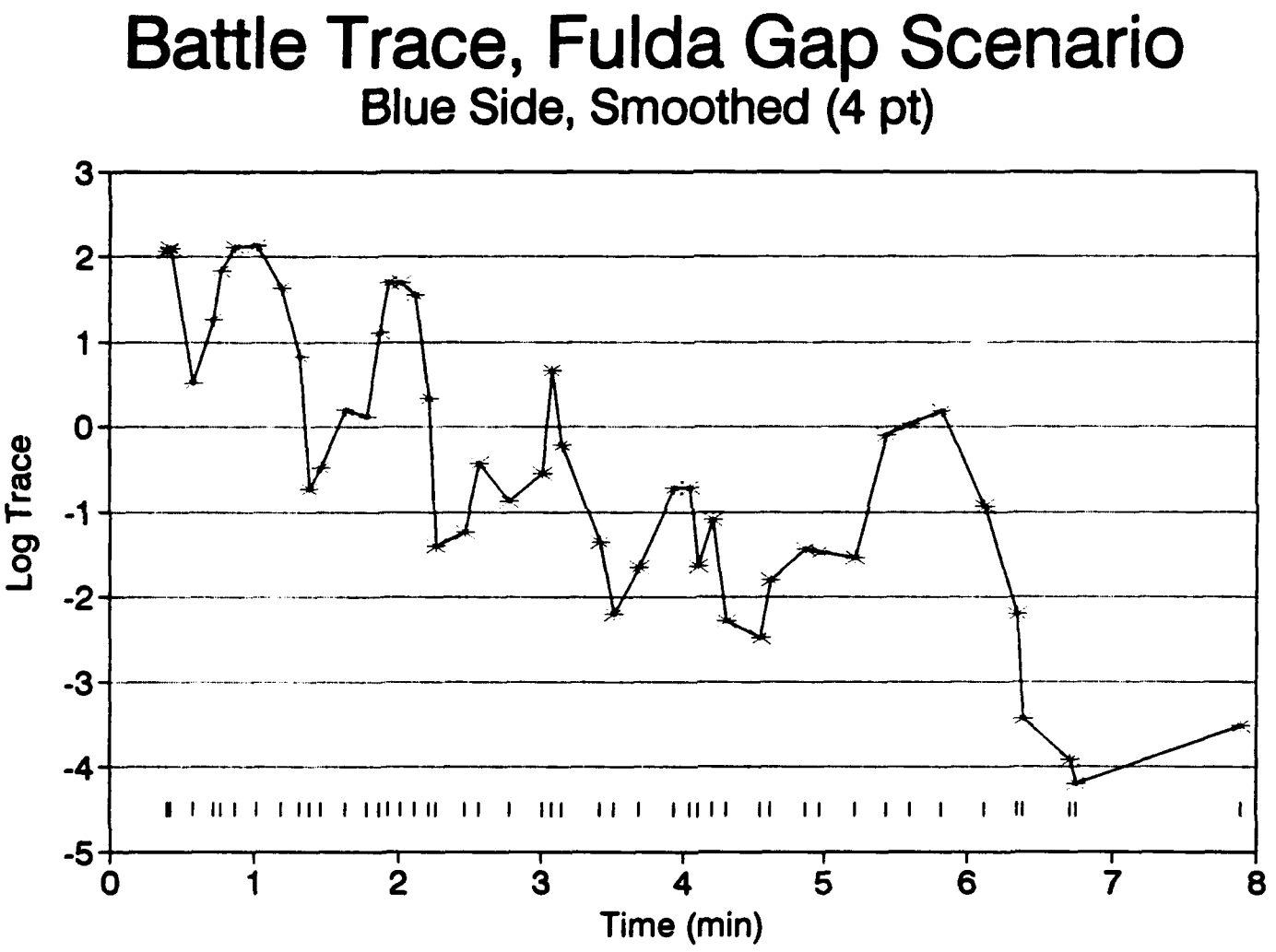


Figure 1. Smoothed log trace for Blue 1 on one iteration of the Fulda Gap scenario. It can be seen that Blue 1 does well initially (B_L>0) but beyond t=2 minutes of battle, the Red forces are generally winning. Tick marks along the bottom of the figure correspond to times at which points on B_L were computed. The threshold used is 32 (i.e., a point is calculated on B_L when approximately 32 shots have been fired by or at Blue 1.

Battle Trace Displays

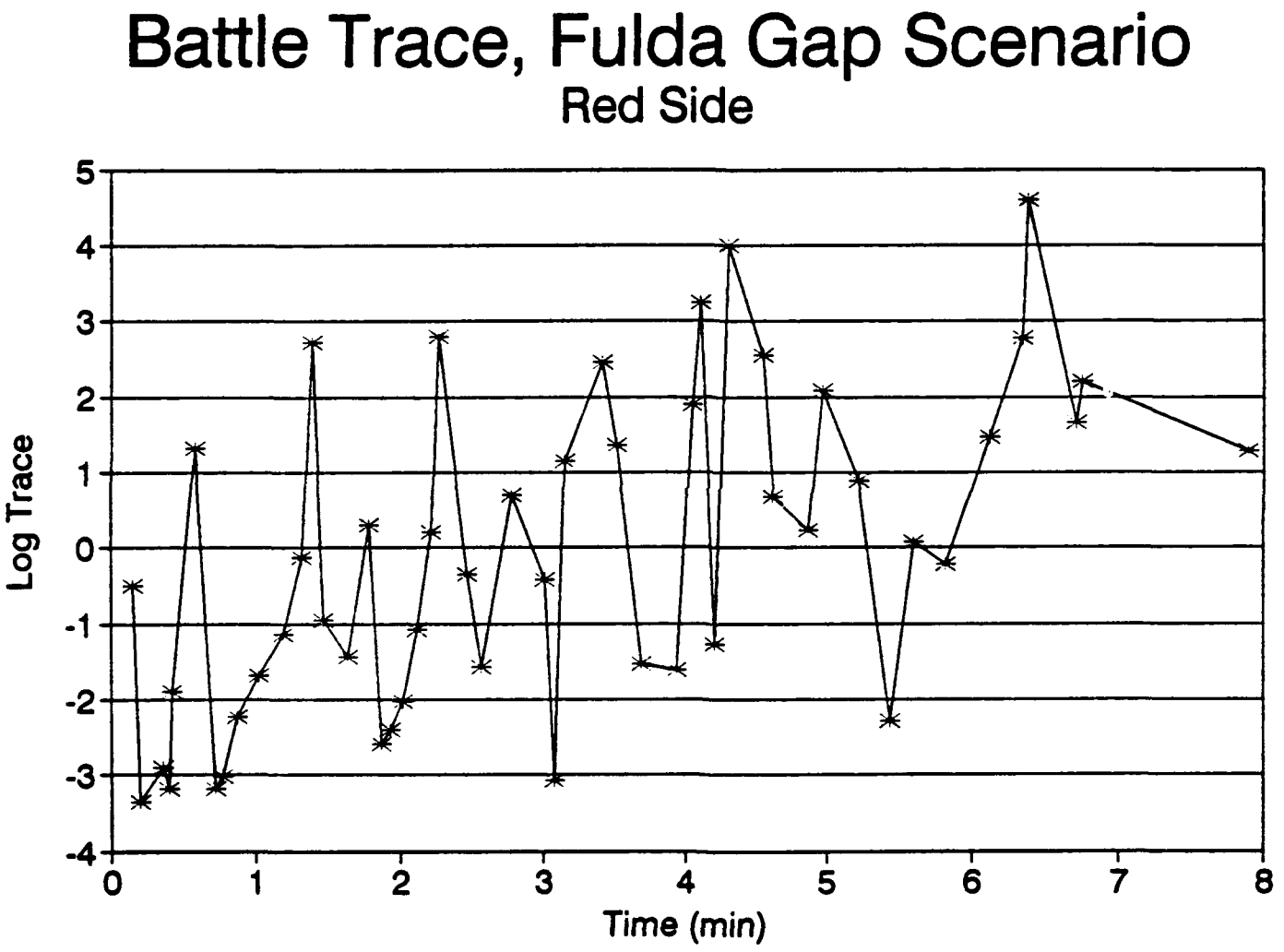


Figure 2. Non-smoothed log trace for all Red forces, corresponding to the iteration shown in Figure 1 (threshold = 32). Note the general improvement in Red's log trace over time; note also the variations in B_L from one time interval to the next.

Battle Trace Displays

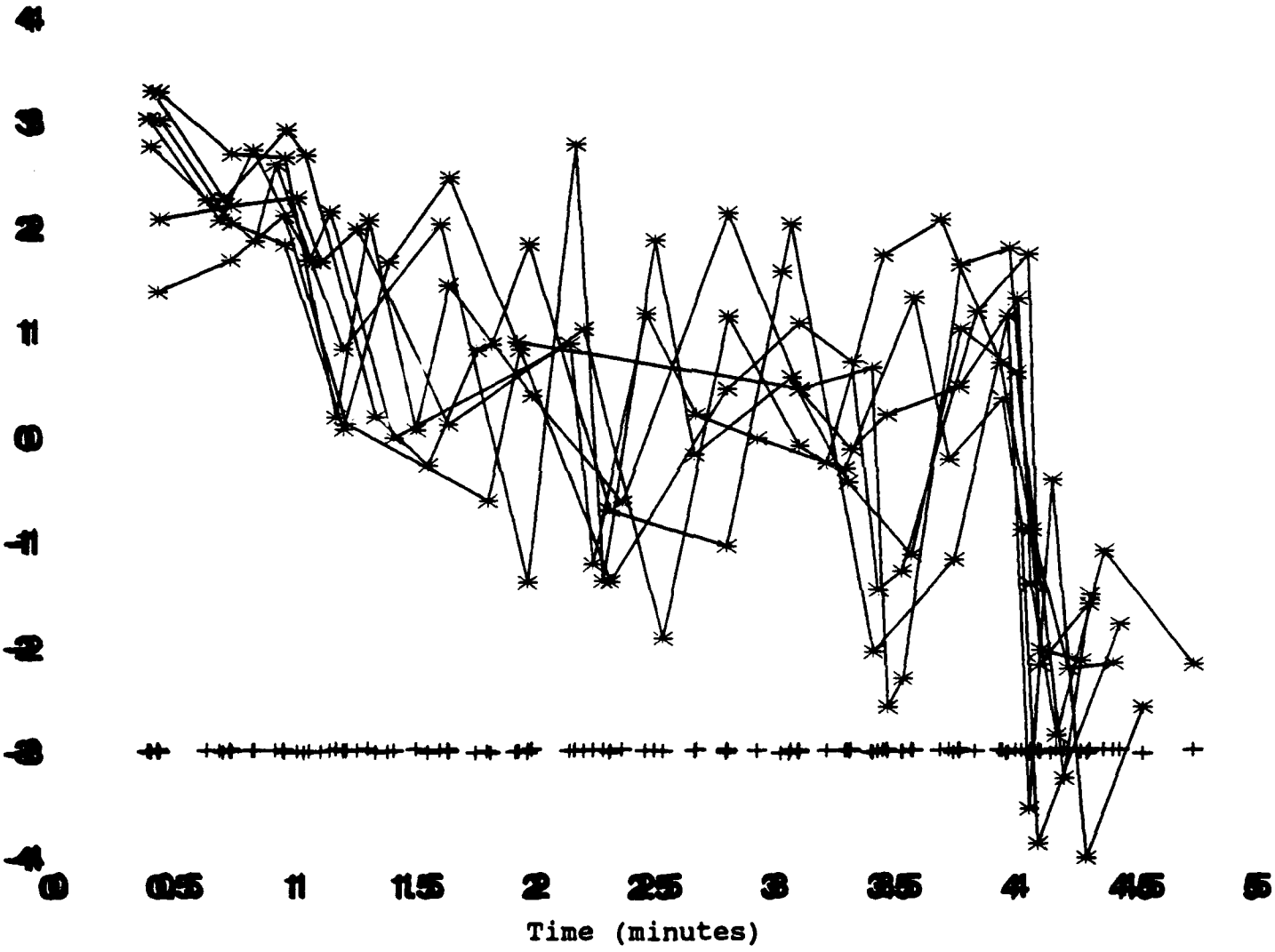


Figure 3. Blue 1 log traces for seven iterations of the Fulda Gap scenario (threshold = 32). Note strong similarity of the log traces in overall shape, despite variations between iterations. The general sharp drop in B_t near t=4 minutes corresponds to a point in the scenario at which Blue 1 starts to move its surviving units. Units on axes appear fuzzy because multiple overlays were used to construct the figure.

Battle Trace Displays

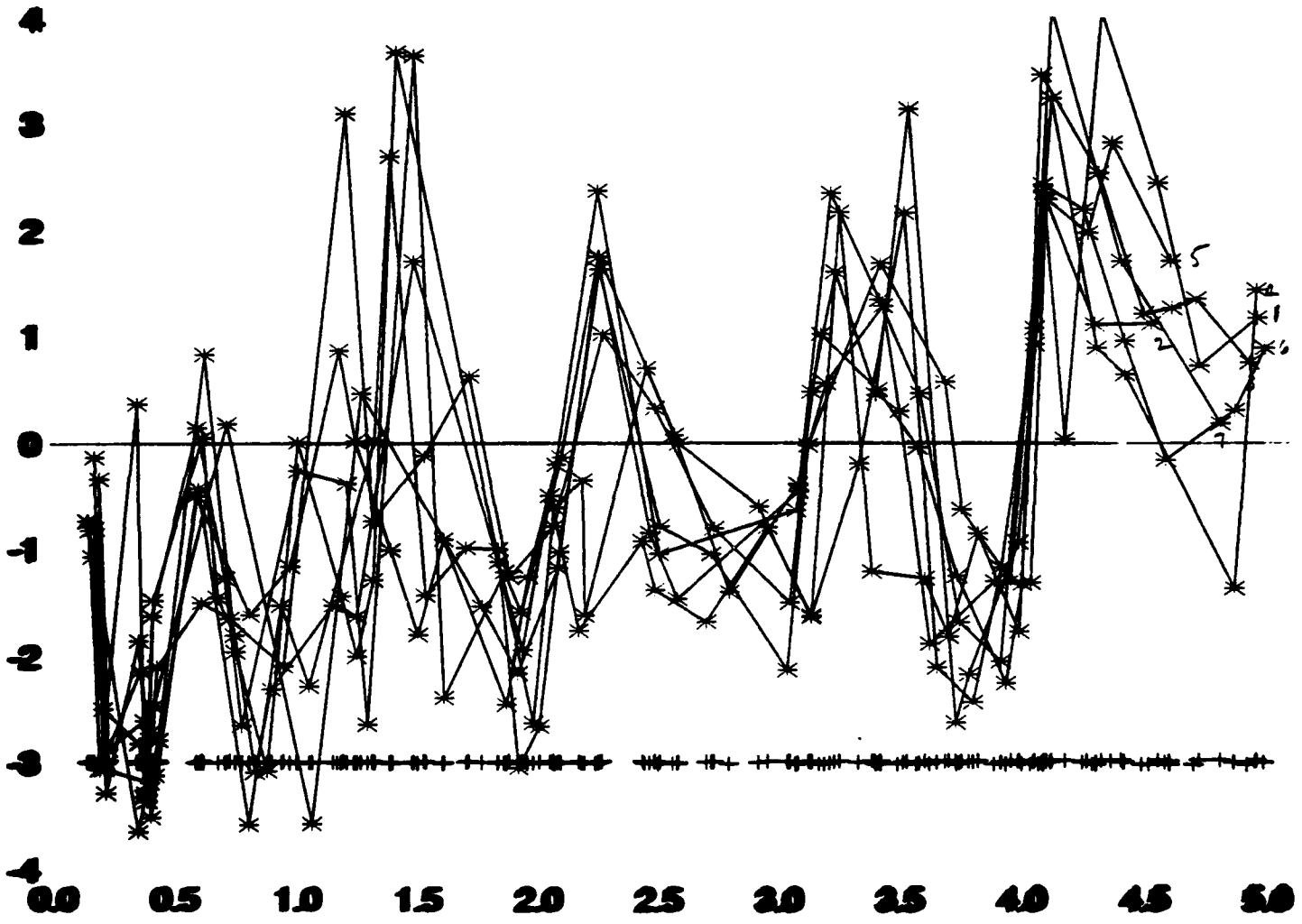


Figure 4. Red task force 1 log traces corresponding to the seven iterations shown in Figure 3.

Battle Trace Displays

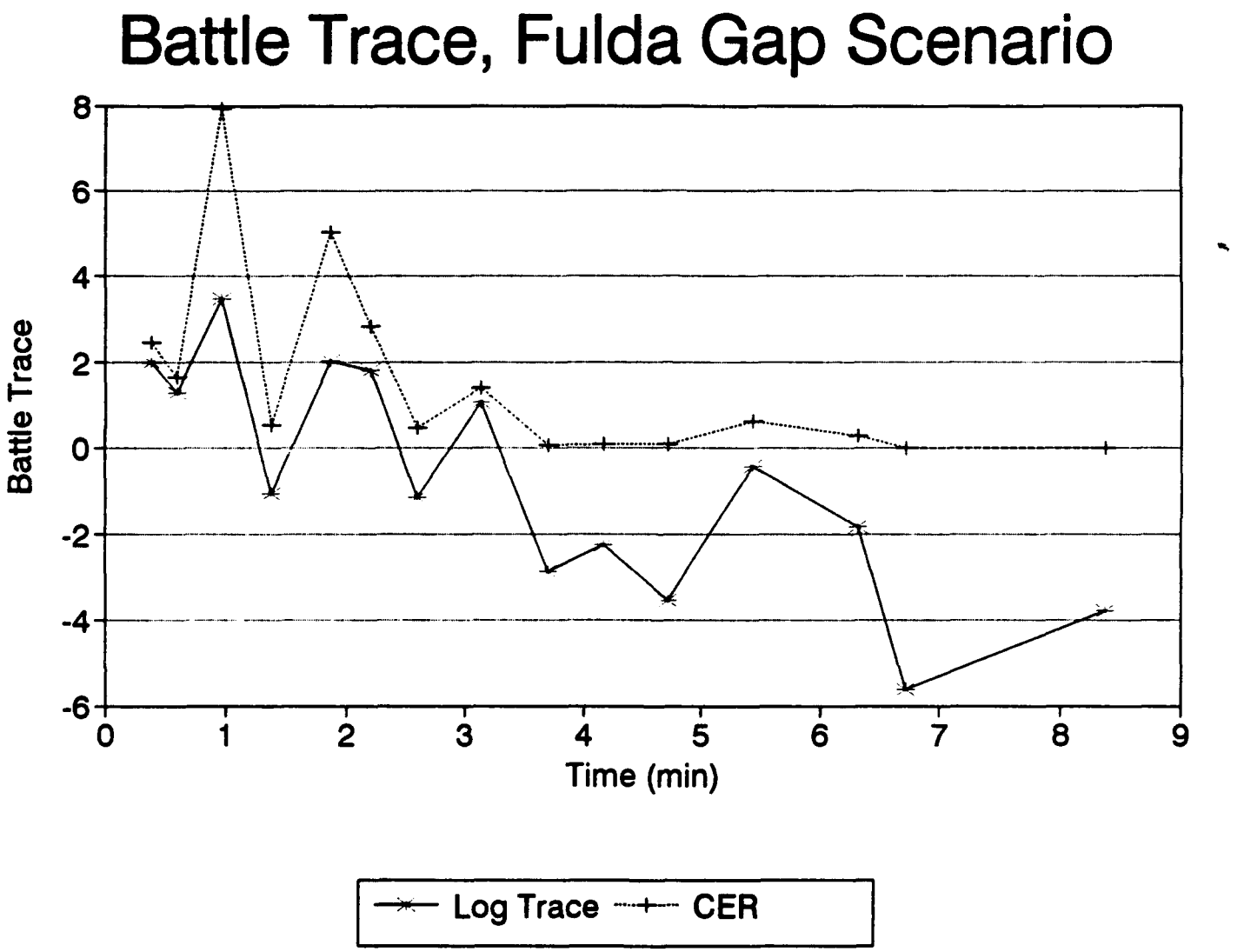


Figure 5. Log trace for Blue 1, together with plot of casualty exchange ratios (CERs) in each time interval. Note similarity of shapes of the two plots.

Battle Trace Displays

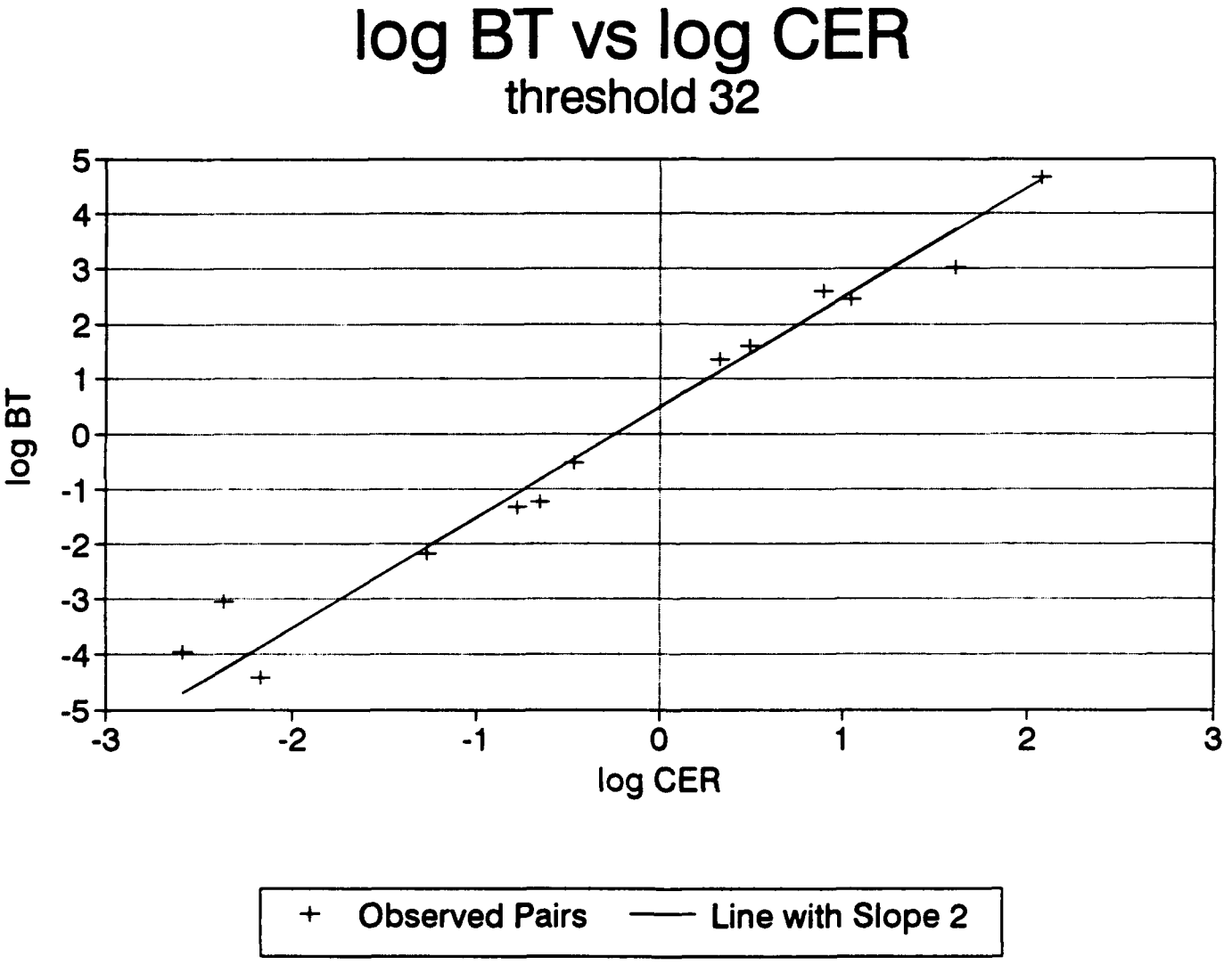


Figure 6. Log-log plot of battle trace versus CER. Note line with slope 2 provides good fit of the plotted points.

Battle Trace Displays

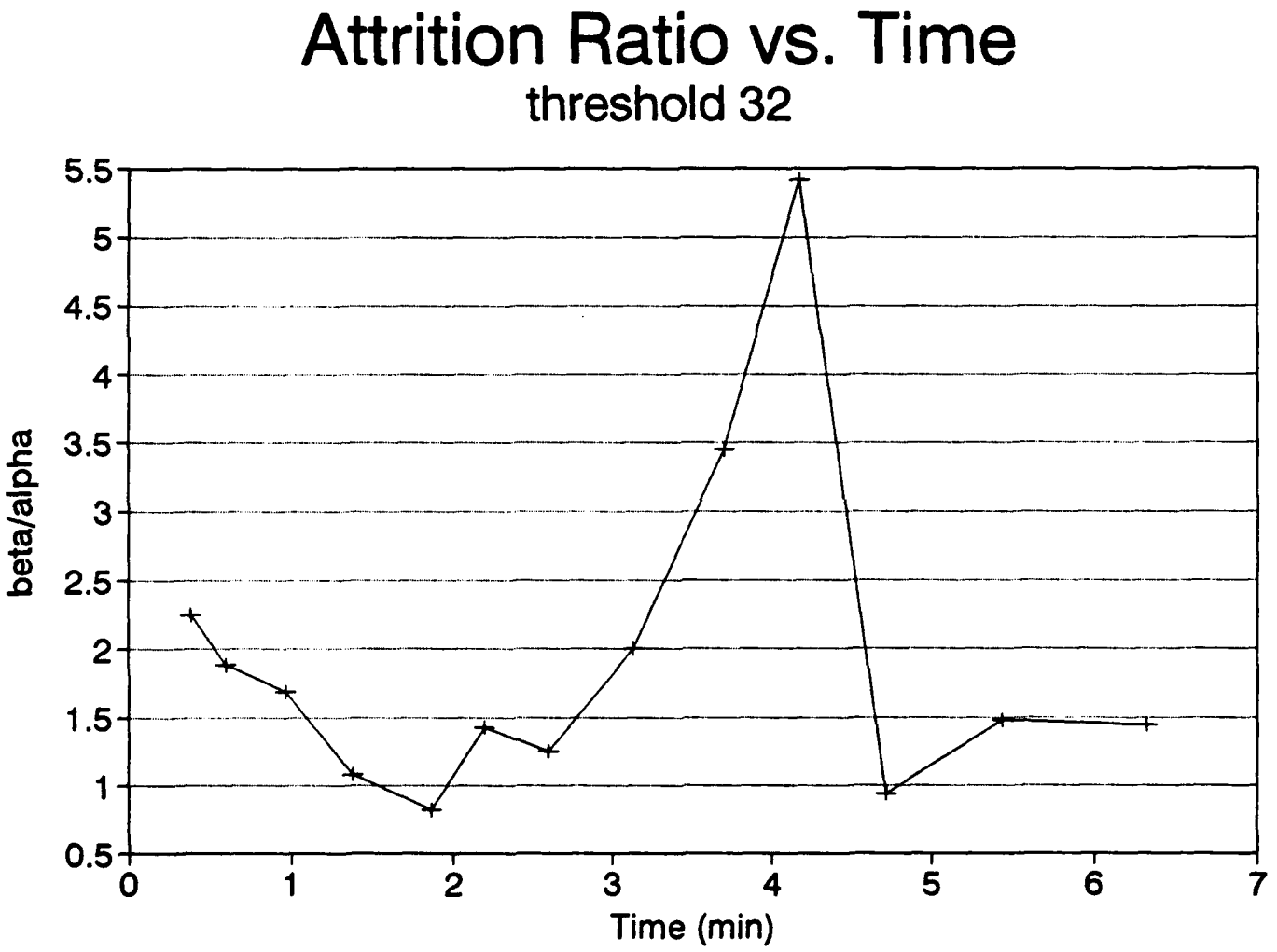


Figure 7. Plot of Blue 1 versus Red force attrition coefficient ratio (β/α) estimates. The values were estimated through residuals relative to the line in Figure 6.

Battle Trace Displays

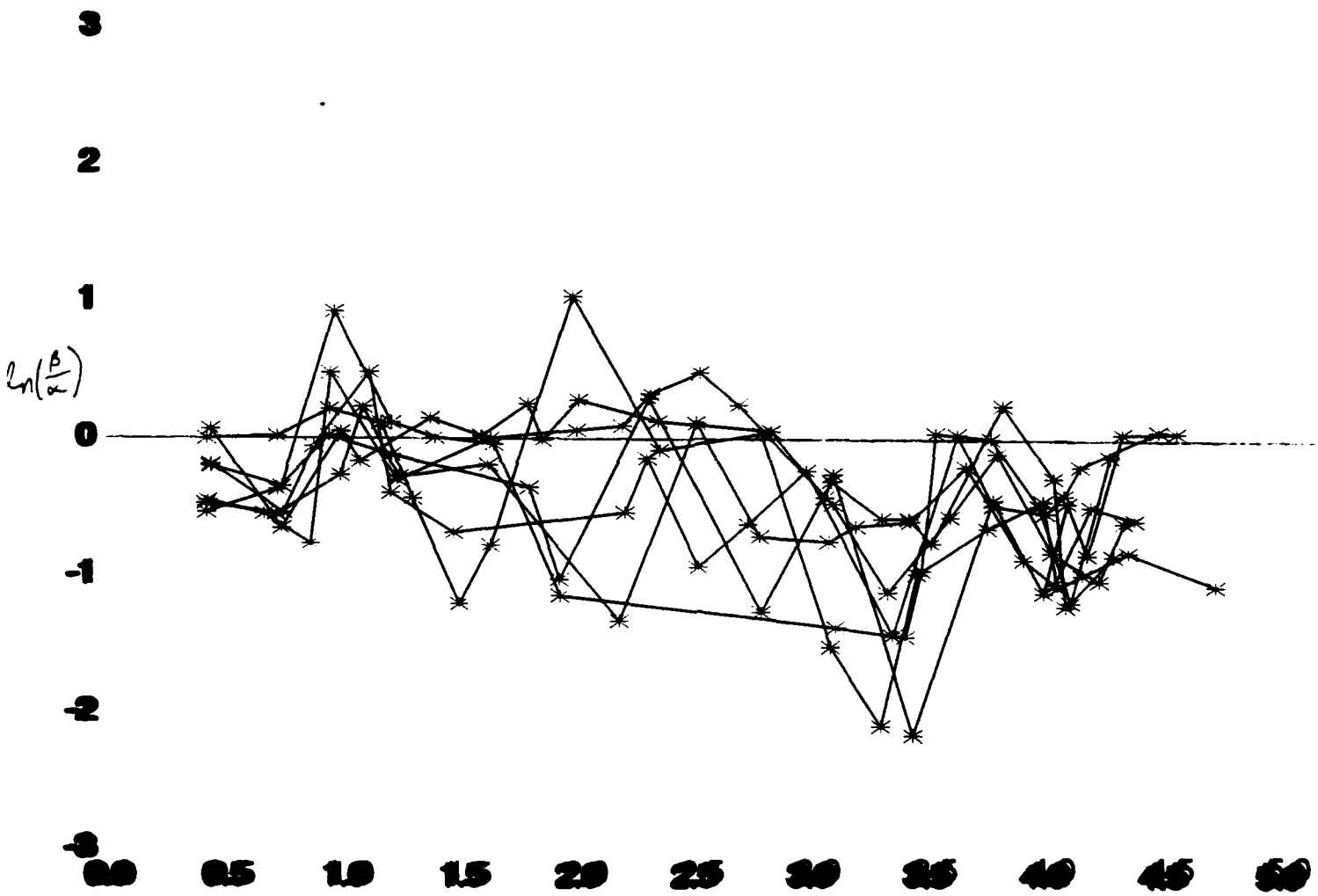


Figure 8. Log Blue 1 versus Red force attrition coefficient ratio estimates, $\ln(\beta/\alpha)$, plotted as a function of time into the battle for seven iterations of the Fulda Gap scenario. These iterations correspond to the battle traces shown in Figure 3.

Battle Trace Displays

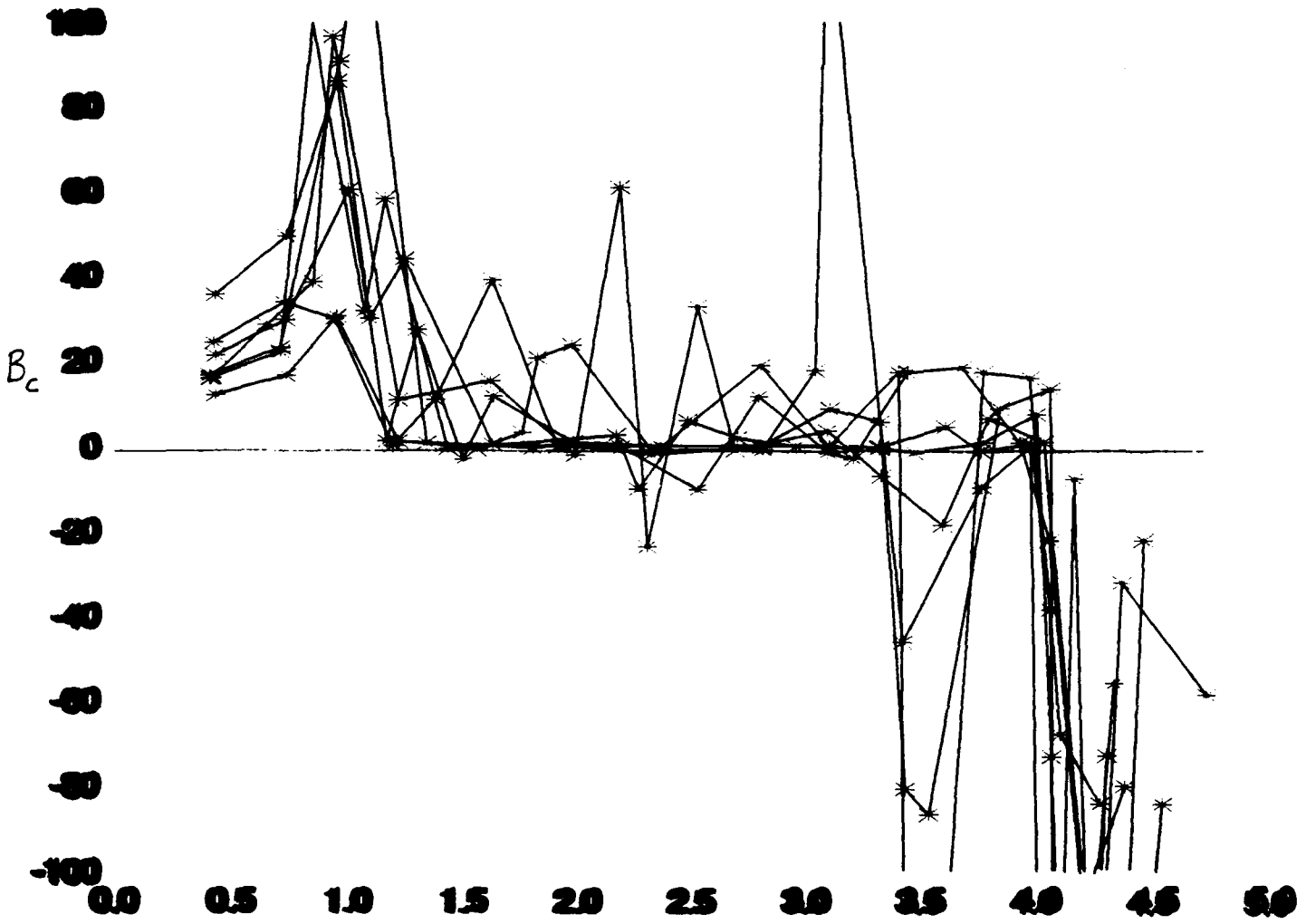


Figure 9. Plots of B_c for seven iterations of the Fulda Gap scenario corresponding to the battle traces shown in Figure 3.

Battle Trace Displays

Appendix 5. Fortran Code for Battle Trace in Janus

Implementation of the battle trace in Janus(A) at TRAC-Monterey required making additions to several subroutines within the Janus Fortran source code. We include listings of the major subroutines affected for the purpose of documentation. As may be seen, the additions have been commented rather generously, so they are "self documented" to a large extent.

```
C------ SUBROUTINE--RELOAD ----------------------------- A.D.KELLNER, TRAC-WSMR

        SUBROUTINE  RELOAD

C*****  MODIFIED FOR BATTLE TRACE COMPUTATION ***** J.C. HOFFMAN, TRAC-MTRY
C                                                   D.R. BARR, NPS
C                                                       2 APR 92

C-------------------------------------------------------------------------C
C                                                                         C
C       PURPOSE:  Sub-driver to process a Direct Fire event. This subrou- C
C                 tine performs TARGET SELECTION for a unit firing a      C
C                 Direct Fire weapon; calls "SHOOT" to simulate/evaluate  C
C                 the firing event; then calls "RLNEXT" to schedule the   C
C                 next Direct Fire event to be processed.                 C
C                                                                         C
C-------------------------------------------------------------------------C
C                                                                         C
C       INPUTS:      ( From COMMON /LOAD/  in  GLOBLOAD.FOR )             C
C                                                                         C
C               NXTUNIT  -  Unit number of shooter                        C
C               NXTSIDE  -  Side of shooter                               C
C                                                                         C
C-------------------------------------------------------------------------C
C                                                                         C
C       DICTIONARY:                                                       C
C                                                                         C
C               IUNIT, ISIDE    -  Unit number, side of shooter           C
C               JUNIT, JSIDE    -  Unit number, side of target            C
C                                                                         C
C               IWPN      -  Absolute Weapon type (1-100)                 C
C               IWPNSLT   -  System Relative Weapon (1-10)                C
C                                                                         C
C-------------------------------------------------------------------------C


        INCLUDE         'JGLOBE:GLOBAL.FOR'     ! CLOCK
        INCLUDE         'JGLOBE:GLOBUNITS.FOR'  ! KSYSTYP,FIRERS,KFIRPRI
        INCLUDE         'JGLOBE:GLBMOPPS.FOR'
        INCLUDE         'JGLOBE:GLBWEAPN.FOR'   ! KGUIDE
        INCLUDE         'JGLOBE:GLOBFIR.FOR'
        INCLUDE         'JGLOBE:GLOBACQ.FOR'
        INCLUDE         'JGLOBE:GLOBLOAD.FOR'
        INCLUDE         'JGLOBE:GLOBMOVE.FOR'   ! IFLYMODE

        INCLUDE         'JGLOBE:GLBTRACE.FOR'   ! BATTLE TRACE Variables***

        DIMENSION       IENEM(NMVISB), PKS(NMVISB), RNGTAR(NMVISB)
        DIMENSION       IWPNSAV(NMVISB), ISTATS(NMVISB), VALUE(NMVISB)


D       IPRINT = 0


        IUNIT  =  NXTUNIT
        ISIDE  =  NXTSIDE


C------- Make popped special-special flyers go through SFRELOAD --------

        IF( ISIDE .EQ. 1 .AND.
     *      FLYERS(KSYSTYP(IUNIT,ISIDE),ISIDE) .EQ. NUMFLYRS  .AND.
     *      IFLYMODE(IUNIT,ISIDE) .LT. 0 ) THEN

          CALL SFRELOAD(IUNIT,ISIDE)
          GOTO 999
```

```
        ENDIF

        IAMFIRNG(IUNIT,ISIDE)  =  0              ! Clear unit's "Firing" flag

        FIRNXT(IUNIT,ISIDE)  =  99999.


D       IF( IFIRST .NE. 99 )  THEN
D       IFIRST  =  99
D       TYPE *
D       TYPE *,'Enter RELOAD unit number for debug print:'
D       ACCEPT *, IPUNIT
D       TYPE *,'Enter RELOAD side for debug print:'
D       ACCEPT *, IPSIDE
D       ENDIF

D       IPRINT = 0
D       IF( (IUNIT.EQ.IPUNIT) .AND. (ISIDE.EQ.IPSIDE) )  IPRINT = 99
D       IF( (IPUNIT .EQ. 0 )  .AND. (ISIDE.EQ.IPSIDE) )  IPRINT = 99

D       IF( IPRINT .GT. 0 )  THEN
D       TYPE *
D       TYPE *,'                  ----------------------------'
D       TYPE *,'                  ---- SUBROUTINE RELOAD ----'
D       TYPE *,'                  ----------------------------'
D       TYPE *
D       ITYPE  =  KSYSTYP(IUNIT,ISIDE)
D       TYPE *,'--- GAME TIME (Sec) =', CLOCK*60.0
D       TYPE *
D       TYPE *,'--- IUNIT,ISIDE,ITYPE =', IUNIT,ISIDE,ITYPE
D       TYPE *,'--- IAMFIRNG(IUNIT,ISIDE) =', IAMFIRNG(IUNIT,ISIDE)
D       END IF



        IF( IDEFL(IUNIT,ISIDE)  .EQ.  2 )     GOTO 300          ! Full defilade
        IF( IDEFL(IUNIT,ISIDE)  .EQ. -2 )     GOTO 300          ! Full defilade
        IF( NSCORE(IUNIT,ISIDE) .LT. 1 )      GOTO 300          ! Dead
        IF( KINOPSTAT(IUNIT,ISIDE) .GT. 0 )   GOTO 300          ! CHEM
        ITYPE  =  KSYSTYP(IUNIT,ISIDE)
        IF( FIRERS(ITYPE,ISIDE) .EQ. 2 )      GOTO 300
        IF( FIRERS(ITYPE,ISIDE) .LE. 0 )      GOTO 300

        IF( MOUNTED(IUNIT,ISIDE) .GT. 0 )  THEN

C---------- Unit is a passenger, update his current status

            IHOST  =  MOUNTED(IUNIT,ISIDE)

            XUNIT(IUNIT,ISIDE)   =  XUNIT(IHOST,ISIDE)
            YUNIT(IUNIT,ISIDE)   =  YUNIT(IHOST,ISIDE)
            SPDU(IUNIT,ISIDE)    =  SPDU(IHOST,ISIDE)
            IHOLFIR(IUNIT,ISIDE) =  IHOLFIR(IHOST,ISIDE)
            TSUPRS(IUNIT,ISIDE)  =  TSUPRS(IHOST,ISIDE)

        ENDIF


        IF( IHOLFIR(IUNIT,ISIDE) .NE. 0 ) THEN
          FIRNXT(IUNIT,ISIDE)  =  CLOCK + 0.2
          GOTO 300
        ENDIF
```

```
          IF( TSUPRS(IUNIT,ISIDE) .GT. CLOCK )  THEN
              FIRNXT(IUNIT,ISIDE)  =  TSUPRS(IUNIT,ISIDE)
              GOTO 300
          ENDIF

          FIRNXT(IUNIT,ISIDE)  =  99999.
          JSIDE  =  3 - ISIDE

          JUNIT   =  0
          SUMVAL  =  0.
          ITGT    =  0

          ITASK   =  KTASKFOR(IUNIT,ISIDE)                ! B-Trace MOE ***
C                                                        !Task Force of Observer

D         IF( IPRINT .GT. 0 )  THEN
D         TYPE *
D         TYPE *,'---------------- LOOK FOR A TARGET...'
D         ENDIF


          DO 100  NM = 1, NMVISB

              IF( KDETECTD(NM,IUNIT,ISIDE) .NE. 1 )  GOTO 100      ! Not detected

              JUNIT  =  NMYUNIT(NM,IUNIT,ISIDE)
              IF( JUNIT                .EQ. 0 )  GOTO 100          ! Empty slot

              JTYPE    =  KSYSTYP(JUNIT,JSIDE)
              JFIRPRI  =  KFIRPRI(ITYPE,JTYPE,ISIDE)

              JTASK    =  KTASKFOR(JUNIT,JSIDE)           !B-Trace MOE ***
                                                       !Task Force of the target

              IF( JFIRPRI .LE. 0              )  GOTO 100          ! No priority
              IF( NSCORE(JUNIT,JSIDE) .LT.  1 )  GOTO 100          ! Already dead

              FIRNXT(IUNIT,ISIDE)  =  CLOCK + 0.1


C---------- Fetch Target range

              DX  =  XUNIT(IUNIT,ISIDE) - XUNIT(JUNIT,JSIDE)       ! From target
              DY  =  YUNIT(IUNIT,ISIDE) - YUNIT(JUNIT,JSIDE)       ! to firer
              RANGE  =  SQRT( DX*DX + DY*DY )


C---------- Check if Target beyond max firing range

              RNGMAX  =  PRIMRNG(ITYPE,ISIDE)
              IF( RANGE .GT. RNGMAX )  GOTO 100



C---------- Check if Target in full defilade

              IF( IDEFL(JUNIT,JSIDE) .EQ.  2  .OR.
     *            IDEFL(JUNIT,JSIDE) .EQ. -2        )  THEN

C-------------- Don't fire unless within 50 meters
                  IF( RANGE .GT. 0.050 )  GOTO 100

              ENDIF
```

```
D       IF( IPRINT .GT. 0 )  THEN
D       TYPE *
D       TYPE *,'---------------- POSSIBLE TARGET:'
D       TYPE *,'--- JUNIT, JSIDE  =', JUNIT,JSIDE
D       TYPE    '--- JTYPE, JPRI   =', JTYPE,JFIRPRI
D       TYPE *,'--- RANGE, RNGMAX =', RANGE,RNGMAX
D       END IF



C---------- Fetch weapon to use against this target

C---------- Force special-special flyers not to use rf missile by setting
C           rounds left to zero

            IFLYTYPE  =  FLYERS(ITYPE,ISIDE)

            IF( ISIDE.EQ.1 .AND. IFLYTYPE.EQ.NUMFLYRS )  THEN
                ISAVRNDS  =  KRLEFT(1,IUNIT,ISIDE)
                KRLEFT(1,IUNIT,ISIDE)  =  0
            ENDIF



            CALL WTCHWPN ( IUNIT,ITYPE,ISIDE,JTYPE,RANGE,  IWPNSLT,IWPN )

D           IF( IPRINT .GT. 0 )  THEN
D           TYPE *,'--- IWPNSLT,IWPN =', IWPNSLT,IWPN
D           END IF



C---------- Restore special-special rf missile rounds previously zeroed.

            IF( ISIDE.EQ.1 .AND. IFLYTYPE.EQ.NUMFLYRS     THEN
                KRLEFT(1,IUNIT,ISIDE)  =  ISAVRNDS
            END IF



C----------- Break if no weapon selected

            IF( IWPNSLT .LE. 0 )  GOTO 100



C---------- Get SSKP for this target

            CALL PROB ( IUNIT, IWPN, JUNIT, ISIDE,  JSIDE,
     *                  RANGE,  DY,   DX,   ISTAT,  SSKP   )

            IF( MOPP(IUNIT,ISIDE) .EQ. 1 )
     *              SSKP  =  SSKP * PHMOPP(IWPN,ISIDE)       ! CHEM

D           IF( IPRINT .GT. 0 )  THEN
D           TYPE *,'--- SSKP =', SSKP
D           END IF

            IF( SSKP .LT. 0.05 )  THEN
                IF( SSKP .GT. 0.0 )  FIRNXT(IUNIT,ISIDE) = CLOCK + 0.02
                GOTO 100
            ENDIF
```

```
C---------- Check for guided weapon that can't track thru smoke

            IF( KGUIDE(IWPN,ISIDE) .EQ. 2 ) THEN

              CALL UNITXYZ ( IUNIT,ISIDE,ITYPE, XF,YF,ZF )

              JTYPE  =  KSYSTYP(JUNIT,JSIDE)
              CALL UNITXYZ ( JUNIT,JSIDE,JTYPE, XT,YT,ZT )

              ISENS  =  1                ! OPTICAL
              DX  =  ABS( XF-XT )
              DY  =  ABS( YF-YT )

              IF( (DX.GT.0.025) .OR. (DY.GT.0.025) )  THEN

                CALL SMOKELOS ( XF,YF,ZF, XT,YT,ZT, ISENS, ISEE )
                IF( ISEE .NE. 1 ) GOTO 100

                CALL DOLASLOS ( XF,YF,ZF, XT,YT,ZT, ISENS, OLEN )
                IF( OLEN .GT. 2.0 ) GOTO 100

              ENDIF

            ENDIF


C---------- Add potential target to local target list

            ITGT  =  ITGT + 1
            IENEM(ITGT)     =  JUNIT
            RNGTAR(ITGT)    =  RANGE
            IWPNSAV(ITGT)   =  IWPNSLT
            ISTATS(ITGT)    =  ISTAT
            PKS(ITGT)       =  SSKP
            VALUE(ITGT)     =  SSKP * FLOAT( JFIRPRI )
            SUMVAL          =  SUMVAL + VALUE(ITGT)

C*****   UPDATE BATTLE TRACE ----------------------                    !BTRACE

C          See if there is a new maximum sspk value for each           !BTRACE
C          enemy task force.  If so, then record the value             !BTRACE
C          in the array BTRACEVAL                                      !BTRACE

               IF ( SSKP .GT. BTRACEVAL(ISIDE,IUNIT,JTASK) ) THEN      !BTRACE
                       BTRACEVAL(ISIDE,IUNIT,JTASK) = SSKP             !BTRACE
               ENDIF                                                   !BTRACE

C*****   END OF BATTLE TRACE UPDATE ---------------                    !BTRACE

  100   CONTINUE



C------ Skip if no valid targets

        IF( ITGT .EQ. 0 )  GOTO 300


C------ If only one target,  select it

        IF( ITGT .EQ. 1 )  THEN
            JUNIT    =  IENEM(ITGT)
            SSKP     =  PKS(ITGT)
            RANGE    =  RNGTAR(ITGT)
            IWPNSLT  =  IWPNSAV(ITGT)
```

```
            ISTAT    =  ISTATS(ITGT)
            GOTO 200
          ENDIF



C-------------- Target with highest value ( SSKP * JFIRPRI )
C-------------- has higest probability of being selected

          CUMRAT = 0.0
          CALL UNIRAN (DRAW)
          CHECK = DRAW * SUMVAL

D         IF( IPRINT .GT. 0 )  THEN
D         TYPE *
D         TYPE *,'DRAW, CHECK =', DRAW,CHECK
D         END IF

          DO 150  I = 1, ITGT

              CUMRAT  =  CUMRAT + VALUE(I)

D             IF( IPRINT .GT. 0 )  THEN
D             TYPE *,'-----LOOP I, CUMRAT =', I,CUMRAT
D             END IF

              IF( CUMRAT .LE. CHECK )  GOTO 150
              JUNIT   =  IENEM(I)
              SSKP    =  PKS(I)
              RANGE   =  RNGTAR(I)
              IWPNSLT =  IWPNSAV(I)
              ISTAT   =  ISTATS(I)
              GOTO 200

  150     CONTINUE


C------ A target has been selected

  200     CONTINUE

D         IF( IPRINT .GT. 0 )  THEN
D         TYPE *
D         TYPE *,'RELOAD HAS SELECTED TARGET UNIT =', JUNIT
D         TYPE *,'--------- SSKP =', SSKP
D         END IF

C*****    UPDATE BATTLE TRACE ----------------------              !BTRACE

C         Update the influence of all shots fired on both         !BTRACE
C         friendly and enemy taskforces.                          !BTRACE


C         Influence of the shot on the TARGET Task Force          !BTRACE

          SIPCTPKS(JSIDE,JTASK) = SIPCTPKS(JSIDE,JTASK) + SSKP    !BTRACE
          KNUMSHOTS(ISIDE,ITASK) = KNUMSHOTS(ISIDE,ITASK) + 1     !BTRACE
          KNUMSHOTS(JSIDE,JTASK) = KNUMSHOTS(JSIDE,JTASK) + 1     !BTRACE

C         The array KNUMSHOTS is zeroed each time the Battle      !BTRACE
C         is computed for a given task force.  This is done       !BTRACE
C         in subprogram BTCOMP called in RUNJANUS                 !BTRACE
```

```
C       Influence of the shot on the FIRER's Total Force          !BTRACE

        DO 210 I = 1, NUMTASKS                                    !BTRACE
          IF ( BTINFL(JSIDE,JUNIT,I) .GT. 0 ) THEN                !BTRACE
               SSHOTPKS(ISIDE,I) = SSHOTPKS(ISIDE,I) + SSKP       !BTRACE
          ENDIF                                                   !BTRACE
  210   CONTINUE                                                  !BTRACE


C*****  END OF BATTLE TRACE UPDATE                                !BTRACE

        IAMFIRNG(IUNIT,ISIDE)  =  1            ! Set unit's "Firing" flag

        CALL SHOOT ( IUNIT, ISIDE, ITYPE, IWPNSLT,
     *               JUNIT, SSKP,  RANGE, ISTAT, KILLS  )




  300   CONTINUE

C------ Schedule next firing event & RETURN

        CALL RLNEXT



D       IF( IPRINT .GT. 0 )  THEN
D       TYPE *
D       TYPE *,'--------------------- END RELOAD ---------------------'
D       TYPE *,'--- FIRNXT    =', FIRNXT(IUNIT,ISIDE) * 60.
D       TYPE *,'--- IAMFIRNG =', IAMFIRNG(IUNIT,ISIDE)
D       TYPE *,'--- TNEXT(4) =', TNEXT(4) * 60.
D       TYPE *
D       TYPE *
D       END IF


  999   CONTINUE

        RETURN
        END
```

```
C----- SUBROUTINE--BTCOMP-----------------D.R. BARR, DEPT OF MATH, NPS
C                                         J.C. HOFFMAN, TRAC-MTRY
C                                              6 APR 92
       SUBROUTINE BTCOMP


C-----  This subroutine sets up Battle Trace data structures needed
C       to compute the Battle Trace for all Janus(A) Task Forces.

C********************** LOCAL VARIABLES **********************************C
C                                                                           C
C       ISIDE -          Index of friendly side                             C
C                                                                           C
C       IUNIT -          Index of friendly unit                             C
C                                                                           C
C       ITASK -          Index of friendly task force                       C
C                                                                           C
C       JTASK -          Index of enemy task force                          C
C                                                                           C
C       PKTASKMAX -      The maximum single shot kill probability           C
C                        recorded against each enemy task force             C
C                                                                           C
C       PKUNITMAX -      The sum of BTRACE over all units assigned to       C
C                        a given friendly task force                        C
C                                                                           C
C***************************************************************************C

       INCLUDE           'JGLOBE:GLOBAL.FOR'
       INCLUDE           'JGLOBE:GLOBUNITS.FOR'
       INCLUDE           'JGLOBE:GLBTRACE.FOR'    !Battle Trace Variables

       REAL              PKTASKMAX(NUMSIDES,NUMTASKS),
     +                   PKUNITMAX(NUMSIDES,NUMTASKS)

C      Zero influence accumulators

       DO 50 I = 1, NUMSIDES
         DO 40 J = 1, NUMTASKS
           PKTASKMAX(I,J) = 0.0
           PKUNITMAX(I,J) = 0.0
  40     CONTINUE
  50   CONTINUE



C-----  Examine each value for the maximum SSKP recorded for each
C       unit on each side.  Determine the maximum SSKP recorded for all
C       enemy task forces.  This value represents the contribution of
C       each particular unit to the battle.
C
C       Next, compute the task force contributions for both friendly and
C       enemy task forces.
C              a.  Sum the contributions for each friendly task force
C       which has reach the THRESHOLD for Battle Trace computation and
C       reinitialize the BTMAX values to zero (This last action has the
C       effect of starting a new computation cycle for the Battle
C       Trace.)
C              b.  Compute the contribution of all friendly units which
C       detect at least one element of each enemy task force for those
C       enemy task forces which have reached the THRESHOLD for battle trace
C       computation.  Do this by summing BTRACEVAL for all friendly units
C       indexed by the appropriate enemy task force.  These sums are stored
C       in array PKUNITMAX.  Reinitialize the appropriate entries in
C       BTRACVAL by setting to zero all entries which correspond to units
C       which contribute to the computation of battle trace for a
C       particular enemy task force.
```

```
      DO 300 ISIDE = 1, NUMSIDES

        JSIDE = 3 - ISIDE

        DO 200 IUNIT = 1, KNUMUNITS(ISIDE)

          DO 100 JTASK = 1, NUMTASKS

C----   Determine maximum SSKP for each friendly unit against
C       all enemy task forces

              IF ( BTRACEVAL(ISIDE,IUNIT,JTASK) .GT.
     *                                  BTMAX(ISIDE,IUNIT) ) THEN
                 BTMAX(ISIDE,IUNIT) = BTRACEVAL(ISIDE,IUNIT,JTASK)
              ENDIF

C----   Compute the contribution for friendly units which observe at least
C       one unit of an enemy task force for those enemy task forces which
C       have reached the threshold for battle trace computation.

              IF ( KNUMSHOTS(JSIDE,JTASK) .GE. THRESHOLD ) THEN
                 PKUNITMAX(JSIDE,JTASK) = PKUNITMAX(JSIDE,JTASK) +
     *                BTRACEVAL(ISIDE,IUNIT,JTASK)

C----   Reinitialize the contribution for each friendly unit which
C       contributed to the calculation of battle trace for an enemy
C       task force.

                 BTRACEVAL(ISIDE,IUNIT,JTASK) = 0
              ENDIF

  100       CONTINUE

          ITASK = KTASKFOR(IUNIT,ISIDE)

C----   Sum the maximum contributions to the battle trace if the battle
C       trace is to be computed for a friendly task force.

C-------------------------------------------------------------------------C
C       Compute the max reduction over columns of BTRACEVAL.  PKTASKMAX C
C       now contains the maximum single shot kill probability of all    C
C       units of a particular friendly task force who detect (and       C
C       have a possibility of firing upon) at least one unit of an      C
C       opposing task force                                             C
C-------------------------------------------------------------------------C

          IF ( KNUMSHOTS(ISIDE,ITASK) .GE. THRESHOLD ) THEN
            PKTASKMAX(ISIDE,ITASK) = PKTASKMAX(ISIDE,ITASK) +
     *                                     BTMAX(ISIDE,IUNIT)

C----   Reinitialize max contribution for next computation interval
C       (NOTE:  The interval of computation in terms of battle time
C       is a function of the dynamics of the scenario and the value
C       set in THRESHOLD.  Larger values of THRESHOLD will result
C       in a longer period of battle time between subsequent calculation
C       of the battle trace.  Likewise, battle periods which do not contain
C       a sufficient amount of weapon firings or shots received by a
C       particular task force will not attain the THRESHOLD for battle
C       trace computation.  The Battle Trace only has meaning for
C       battle periods which contain weapon effects related interactions
C       between combatants.)

            BTMAX(ISIDE,IUNIT) = 0

          ENDIF
```

```
  200     CONTINUE
  300   CONTINUE

C-----  Compute the Battle Trace for All Task Forces where threshold
C               conditions are met

        DO 500 ISIDE = 1, NUMSIDES

          DO 400 ITASK = 1, NUMTASKS
          IF ( KNUMSHOTS(ISIDE,ITASK) .GE. THRESHOLD ) THEN

            CALL BTCOMPUTE(ISIDE,ITASK,PKUNITMAX,PKTASKMAX)

C-----  Reinitialize battle trace by setting accumulator for shots fired
C       and recieved to zero after battle trace is computed for a
C       particular task force.
            KNUMSHOTS(ISIDE,ITASK) = 0
C-----  Reinitialize the accumulators of the influence of shots in terms
C       of SSKP

            SIPCIPKS(ISIDE,ITASK)  = 0
            SSHOTPKS(ISIDE,ITASK)  = 0
C-----
          ENDIF

  400     CONTINUE

  500   CONTINUE

        RETURN
        END
```

```
C----- SUBROUTINE BTCOMPUTE ---------------D.R. BARR, DEPT OF MATH, NPS
C                                          J.C. HOFFMAN, TRAC-MTRY
C                                               7 APR 92

      SUBROUTINE BTCOMPUTE( ISIDE, ITASK, PKUNITMAX, PKTASKMAX )

C-------------------------------------------------------------------C
C                                                                   C
C       INPUTS:                                                     C
C               ISIDE-  The side of the friendly task force.        C
C                                                                   C
C               ITASK-  The task force of the friendly side for which C
C                       the battle trace is to be computed.         C
C                                                                   C
C               PKUNITMAX -  An array with dimentions NUMSIDES,NUMTASKS C
C                       which contains the sum of the SSKP values which C
C                       constitute the contribution of all enemy units C
C                       to the Battle Trace measure of effectiveness. C
C                                                                   C
C               PKTASKMAX -  An array with dimensions NUMSIDES,NUMTASKS C
C                       which contains the sum of the SSKP values   C
C                       which represent the capability of a friendly C
C                       task force to contribute to the battle.     C
C                                                                   C
C               BTMETH -     An integer which designates the method C
C                       to be used in the computation of Battle Trace. C
C                       This value is a user input obtained from    C
C                       CITY Janus Screen IV (Form 1119).           C
C                                                                   C
C                                                                   C
C-------------------------------------------------------------------C

      INCLUDE           'JGLOBE:GLOBAL.FOR'
      INCLUDE           'JGLOBE:GLOBUNITS.FOR'
      INCLUDE           'JGLOBE:GLBTRACE.FOR'

      DIMENSION         PKTASKMAX(NUMSIDES,NUMTASKS),
     +                  PKUNITMAX(NUMSIDES,NUMTASKS)

C---  Get Battle Trace intermediate values

      TYPE *, ' TIME  ISIDE  ITASK   DELTA P    P    DELTA P    P    P-AL'

      R1DUM =  SSHOTPKS(ISIDE,ITASK)
      RDUM  =  SIPCTPKS(ISIDE,ITASK)
      B1DUM =  PKUNITMAX(ISIDE,ITASK)
      BDUM  =  PKTASKMAX(ISIDE,ITASK)


      IF ( BTMETH .EQ. 1 ) GOTO 100    !Compute LOG-TRACE
      IF ( BTMETH .EQ. 2 ) GOTO 200    !Compute Standard Battle Trace
      IF ( BTMETH .EQ. 3 ) GOTO 300    !Compute Symmetric Battle Trace
      IF ( BTMETH .EQ. 4 ) GOTO 300    !Compute Incrmented Standard Trace

      TYPE *, 'BATTLE TRACE NOT COMPUTED ***************'
      RETURN

  100 CONTINUE          !****** COMPUTE LOG-TRACE ******

      BTVAL = LOG(R1DUM+1) - LOG(RDUM+1) - LOG(B1DUM+1) + LOG(BDUM+1)
      GOTO 900

  200 CONTINUE          !****** COMPUTE STANDARD BATTLE TRACE ******

C     CHECK FOR ZERO ELEMENTS
      IF ( (R1DUM .LE. 0) .OR.
```

```
*           (RDUM .LE. 0) .OR.
*           (B1DUM .LE. 0) .OR.
*           (BDUM .LE. 0) )       THEN
      BTVAL = 1
    ELSE
      BTVAL = (R1DUM / RDUM) * (BDUM / B1DUM)
    ENDIF
    GOTO 900


300 CONTINUE          !****** COMPUTE INCREMENTED STANDARD BATTLE TRACE

    BTVAL = ((R1DUM+.01)/(RDUM+.01)) * ((BDUM+.01)/(B1DUM+.01))


C   ****** COMPUTE INCREMENTED SYMMETRIC BATTLE TRACE ******

    IF ( (BTMETH .EQ. 3) .AND. (BTVAL .LT. 1.0) ) THEN
      BTVAL = 2 - (1/BTVAL)
    ENDIF

900 TYPE *, CLOCK,ISIDE,ITASK,R1DUM,RDUM,B1DUM,BDUM,BTVAL

    RETURN

    END
```

```
C----- SUBROUTINE--BTVIS ----------------------J.C.HOFFMAN, TRAC-MTRY
C                                                1 APR 92

        SUBROUTINE BTVIS ( IUNIT, ISIDE, KOUNT )

C*****  THIS SUBPROGRAM SUPPORTS BATTLE TRACE MOE RESEARCH        ***** C


C-------------------------------------------------------------------------C
C                                                                         C
C       This subroutine builds a data structure employed in the           C
C       computation of the Battle Trace MOE.                              C
C                                                                         C
C       CALLING ROUTINE -- NRANGE                                         C
C                                                                         C
C       INPUTS:     KOUNT -       Number of enemy units within visibility C
C                                 range                                   C
C                                                                         C
C                   NMYID -       List of enemy unit ID numbers within    C
C                                 visibility range                        C
C                                                                         C
C                   ISIDE -       Side of friendly unit                   C
C                                                                         C
C                   IUNIT -       Unit of friendly unit                   C
C                                                                         C
C                   JSIDE -       Side of enemy unit (from GLOBSRCH)      C
C                                                                         C
C       OUTPUTTS:         Update of logical array BTINFL defined in       C
C                         GLBTRACE.FOR                                    C
C                                                                         C
C       FUNCTION:         For each observer, the appropriate entries      C
C                         are modified in BTINFL to indicate which enemy  C
C                         task forces have at least one unit which        C
C                         is a candidate target for the observer          C
C                                                                         C
C       USE:              BTINFL values are updated every call to the     C
C                         SEARCH subroutine in master Janus scheduling    C
C                         routine found in RUNJANUS                       C
C                                                                         C
C                         BTINFL contains logical data used to compute    C
C                         the values contained in SSHOTPKS in the         C
C                         subroutine RELOAD                               C
C                                                                         C
C-------------------------------------------------------------------------C

        INCLUDE         'JGLOBE:GLOBAL.FOR'
        INCLUDE         'JGLOBE:GLOBUNITS.FOR'          !KTASKFOR
        INCLUDE         'JGLOBE:GLOBSRCH.FOR'           !KOUNT,NMYID
        INCLUDE         'JGLOBE:GLBTRACE.FOR'           !BTINFL


        DO 200 I = 1, NUMTASKS
          BTINFL(ISIDE,IUNIT,I) = 0             !Set flag to zero
  200   CONTINUE

C**  Set logical flag to 1 for each opposing task force which can be
C       seen by the observer (IUNIT)

        DO 300 J = 1, KOUNT
          JUNIT = NMYID(J)
          JTASK = KTASKFOR(JUNIT,JSIDE)
          BTINFL(ISIDE,IUNIT,JTASK) = 1
  300   CONTINUE

        RETURN
```

```
C-------------------------------- FILENAME:  GLOBAL.FOR


      INCLUDE          'JGLOBE:GLBPARAM.FOR'
      INCLUDE          'JGLOBE:GLOBMAIN.FOR'
```

DISTRIBUTION LIST

Commander U.S. Army Training and Doctrine Command ATTN: ATCD Fort Monroe, VA 2361-5000	1
HQDA DUSA-OR ATTN: Mr. Walter W. Hollis Room 2E660, The Pentagon Washington, D.C., 20310-0102	1
Commander U.S. Army TRADOC Analysis Command ATTN: ATRC Fort Leavenworth, KS 66027-5200	1
HQDA ATTN: DAPE-MR Washington, D.C. 20310-0300	3
Director U.S. Army TRADOC Analysis Command-WSMR ATTN: ATRC-W White Sands Missile Range, NM 88002-5502	1
Director U.S. Army TRADOC Analysis Command-WSMR ATTN: ATRC-WSL (Technical Library) White Sands Missile Range, NM 88002-5502	1
Director U.S. Army TRADOC Research Activities ATTN: ATRC-RDM P.O. Box 8692, Monterey, CA 93943-0692	1
Director U.S. Army TRADOC Analysis Command-Ft. Benjamin Harrison ATTN: ATRC-FB Fort Benjamin Harrison, IN 46216-5000	1
Conflict Simulation Center Lawrence Livermore National Laboratory P.O. Box 808 L-315 Livermore, CA 94550	1
U.S. Military Academy Department of Systems Engineering ATTN: COL James Kays West Point, NY 10996	1

U.S. Military Academy 1
Department of Mathematics
ATTN: COL Frank Giordano
West Point, NY 10996

Rand Corporation 1
ATTN:
1700 Main Street
Santa Monica, CA 90406-2138

Director 1
U.S. Army Concepts Analysis Agency
8120 Woodmont Ave.
Bethesda, MD 20814-2797

U.S. Army Combined Arms Research Library (CARL) 1
ATTN: ATZL-SWS-L
Fort Leavenworth, KS 66027

Defense Technical Information Center 2
ATTN: DTIC, TCA
Cameron Station
Alexandria, VA 22314

Prof. David L. Bitters 1
6427 Sagamore Rd.
Shawnee Mission, KS 66208

Prof. Wayne Hughes 1
Code OR/HI
Naval Postgraduate School
Monterey, CA 93943

Prof. Don Barr 10
Department of Systems Analysis
U.S. Military Academy
West Point, NY 10996

Prof. Maurice Weir 1
Code MA/Wc
Naval Postgraduate School
Monterey, CA 93943

Prof. Bard Mansager 1
Code MA/Ma
Naval Postgraduate School
Monterey, CA 93943

Prof. Samuel Parry 1
Code OR/Py
Naval Postgraduate School
Monterey, CA 93943

Mr. James Hoffman 5
8073 West Ken Caryl Circle
Apt. D
Littleton, CO 80123

Commander 1
U.S. Army TEXCOM Experimentation Center
ATTN: ATCT-TE-ST (Mr. West)
Jolon, CA 93928

Director of Research 1
Administration
Code 08
Naval Postgraduate School
Monterey, CA 93943

Library 2
Code 52
Naval Postgraduate School
Monterey, CA 93943

Prof. Richard Franke 1
Code MA/Fe
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943